OPTIMIZED DESIGN AND ENERGY MANAGEMENT OF HEATING, VENTILATING AND AIR CONDITIONING SYSTEMS BY EVOLUTIONARY ALGORITHM

by

KWONG FAI FONG

A thesis submitted

in partial fulfillment

of the award of

Doctor of Philosophy

of

De Montfort University

December 2006

Abstract

In the study of heating, ventilating and air conditioning (HVAC) systems, it is becoming increasingly common to adopt a combined simulation-optimization approach to handle the design and energy management problems. Due to the large variety of mathematical models and the complexity in solving the differential-algebraic equation set, it is normal to use a commercial plant simulation package for modeling purpose. The plant simulation models developed can be a useful tool to study "what if" variations to the system parameters. However if a large number of problem variables are involved, this would be demanding and reliant on the intuition of the operator to consider different scenarios. In this case, suitable optimization techniques can be introduced in order to ease the workload and determine reliable and optimal solution. However due to the typically multimodal, multidimensional, nonlinear, mixed continuous-discrete, and highly constrained nature of HVAC problems, this renders to the traditional analytical and gradient-based optimization methods not effective in these cases. As a result, other numerical and heuristic optimization methods have been considered. Among such methods, evolutionary algorithm (EA) has been investigated for HVAC problems by researchers in the last decade. Although there are primarily three paradigms in EA – genetic algorithm, evolutionary programming and evolution strategy – only genetic algorithm has been widely applied and developed for engineering applications, but few studies were concerned with the other two paradigms for HVAC problems.

In this research, a plant simulation model of HVAC system was developed using the TRNSYS simulation environment. An optimization platform, referred to EA Suite, was established using MATLAB based on the paradigms of evolutionary programming and evolution strategy. The EA Suite included the major EA operators of mutation, selection, recombination and constraint handling, as well as a coupling linkage with the TRNSYS for simulation-optimization functionality. In-depth and systematic qualitative studies were carried out to understand the performances of different choices and combinations of EA operators, in order to develop a robust EA (REA). A major problem in analysis of HVAC simulation problems was the length of time taken to evaluate the objective function by calling the related external simulation program. This meant that the number of evaluation function calls was a very significant factor in determining the search efficiency, which was not normally the case in optimization research. Finally, by analyzing a number of local HVAC design and energy management optimization problems, the REA was demonstrated to be effective in tackling real-life engineering applications where there would be limited evaluation function calls.

Acknowledgements

The author would like to give my hearty gratitude to my first supervisor Professor Vic Hanby, not just for his continual inspiration and guidance to my research works all along these five years of part-time study, but also for his life impact of the enthusiasm and commitment in research. I would also thank for his proof reading this thesis. I would like to give thanks to my second and local supervisor Dr. T.T. Chow for his continual monitoring of my research progress, so that I would not be deflected from the priority of study in the demanding daily working life, and keep on the momentum of study as a part-time research candidate.

I would like to thank my second supervisor Dr. John Mardaljevic for his advice at the start of my study. I would also like to thank Dr. Jonathan Wright for his valuable advices and comments on the role of constraint handling techniques and selection in evolutionary algorithm. Thanks also to Mr. Apple Chan, who has regular in-depth discussions in the course of my research works. Also thanks to Dr. Kelvin Yuen, for the fruitful discussions about the current research development of analytical study of genetic and evolutionary algorithm.

The last but not the least, I would like to give my hearty gratitude to my beloved wife Ivy and two daughters Anna and Joyce. Without their prayers and support, I could not maintain my spirit in facing different challenges of the research works throughout these five years of part-time study.

CONTENTS

List of Tables

List of Figures

List of Symbols and Abbreviation

C	constraint function
c	"cooling rate" for simulated annealing, problem-specific scalar for particle swarm optimization, raw constraint value; or abbreviation of constraint handling for performance graph
C(0,1)	Cauchy random number with mean = 0 and variance = 1
C_{pa}	specific heat capacity of moist air at constant pressure (kJ $kg^{-1}K^{-1}$)
C_{pcw}	specific heat capacity of sea water at constant pressure (kJ $kg^{-1}K^{-1}$)
C_{pw}	specific heat capacity of water at constant pressure (kJ $kg^{-1}K^{-1}$)
C_s	average slope of saturation air enthalpy versus temperature (kJ $kg^{-1}K^{-1}$)
CH	option of constraint handling operator
COP_{nom}	coefficient of performance at nominal cooling capacity
$COP_{nom,n}$	nominal coefficient of performance of the n^{th} chiller
COP_{rated}	coefficient of performance at rated cooling capacity
COP_{ratio}	ratio of COP_{nom} to COP_{rated}
D	diameter of round duct (m)
e	abbreviation of epoch of termination
E_{aheat}	electricity consumption of centralized auxiliary electric heater (kJ)
E_{AHU}	energy consumption of AHU fan (kJ)
$E_{c,g}$	unit electrical energy cost in operation mode g ($/kWh)
E_{CHP}	energy consumption of chilled water pumps (kJ)
E_d	energy demand cost ($/kWh)
E_{eheat}	electricity consumption of conventional electric heating (kJ)
E_{FCF}	energy consumption of free cooling fan (kJ)
E_{OAF}	energy consumption of outdoor air fan (kJ)
E_p	first year energy cost ($)
E_{pp}	electricity consumption of circulation pump set (kJ)
E_{RAF}	energy consumption of return air fan (kJ)
E_s	initial cost for round/rectangular ducts ($)
E_{saving}	year-round saving in electricity consumption by using solar water heating system against conventional electric heating (kJ)
E_{sheat}	electricity consumption of solar heating system (kJ)
E_{SWP}	energy consumption of sea water pumps (kJ)
E_{SWP}	energy consumption of sea water pumps (kJ)
E_{total}	total energy consumption of entire HVAC system (kJ)
EA	evolutionary algorithm

$epoch_{max}$	epoch of termination
F	present worth owning and operating cost ($); or total power consumption of whole heat rejection loop (kW)
$\mathfrak{I}$	feasible region for constrained problem
f	objective function value
f_{best}	optimal objective function value at $epoch_{max}$
$f_{best,avg}$	average of optimal objective function value in multiple runs
$f_{best,last}$	optimal objective function value at the last run
$f_{best,stdev}$	standard deviation of optimal objective function value in multiple runs
f_{dp}	fitness value by using constraint handling method of dynamic penalty
f_{max}	maximum objective function value of population by using constraint handling method of infeasibility discrimination
f_{mean}	mean objective function value at $epoch_{max}$
$f_{mean,avg}$	average mean objective function value in multiple runs
f_o	fitness of offspring in simulated annealing
F_p	penalty-based evaluation function
f_p	fitness of parent in simulated annealing
f_{sel}	fitness value by using constraint handling method of infeasibility discrimination
G	inequality constraint function; or number of different system operation modes
g	violated constraint value; or operation mode for duct design optimization problem = 1 for low flow and non-peak utility rate; = 2 for high flow and non-peak utility rate; = 3 for high flow and peak utility rate; or = 4 for low flow and peak utility rate.
G(0,1)	Gaussian random number with mean = 0 and variance = 1
g_{best}	location vector for particle swarm optimization
g_{last}	constraint violation of constraint function of elite individual at the last run
g_{max}	maximum violated constraint value
G_{nom}	nominal cooling capacity of chiller (kW)
G_{rated}	rated cooling capacity of chiller (kW)
G_{ratio}	ratio of G_{nom} to G_{rated}
g_{total}	total constraint violation
$g_{total,avg}$	average of total constraint violation of elite individual at $epoch_{max}$ in multiple runs (constrained problem only)

$g_{total,last}$	total constraint violation of elite individual at the last run
$g_{total,stdev}$	standard deviation of total constraint violation of elite individual at $epoch_{max}$ in multiple runs (constrained problem only)
GA	genetic algorithm
H	equality constraint function; or duct height (m)
h_a	specific enthalpy of moist air (kJ kg^{-1})
h_s	specific enthalpy of saturated air (kJ kg^{-1})
HVAC	heating, ventilating and air conditioning
I^+	positive integer set
infy	infeasibility
j	number of operating condenser water pump, j = 1, 2 or 3
K	parameter of stochastic strategy parameter
k	number of operating cooling tower fan, k = 1, 2 or 3
L	chromosome length; or duct length (m)
lb	lower bound
m	number of inequality constraint functions; or abbreviation of mutation for performance graph
m*	ratio of air to water effective capacitance rate for wet analysis
m_a	mass flow rate of dry air (kg s^{-1})
$m_{a,k}$	mass flow rate of the k^{th} cooling tower fan (kg s^{-1})
$m_{a,nom,k}$	nominal mass flow rate of the k^{th} cooling tower fan (kg s^{-1})
m_{cw}	mass flow rate of cooling water (kg s^{-1})
m_{dhw}	mass flow rate of domestic hot water (kg hr^{-1})
m_{pp}	mass flow rate of circulation pump set (kg hr^{-1})
m_w	mass flow rate of chilled water (kg s^{-1})
m_{wj}	mass flow rate of the j^{th} condenser water pump (kg s^{-1})
$m_{w,nom,j}$	nominal mass flow rate of the j^{th} condenser water pump (kg s^{-1})
MGA	micro-genetic algorithm
MU	option of mutation operator; or
n	abbreviation of number of population; or number of operating chiller, n = 1, 2 or 3
n_{con}	number of constraint functions
n_{eval}	number of evaluation function calls
n_{pop}	number of population
n_{var}	number of problem variables

n_{viol}	number of violated constraints
NTU	overall number of transfer units
p	penalty function
p_a	annealing probability for simulated annealing
p_{best}	location vector for particle swarm optimization
P_c	cumulative distribution function of Cauchy probability density function
p_c	Cauchy probability density function
p_f	probability to proceed stochastic ranking for not both feasible individuals
$P_{fan,nom,k}$	nominal power of the k^{th} cooling tower fan (kW)
p_g	Gaussian (normal) probability density function
P_g^s	maximum subsystem path pressure during operation mode g (Pa)
P_{nom}	nominal chiller power (kW)
$P_{pump,nom,j}$	nominal power of the j^{th} condenser water pump (kW)
$P_{t,g}$	path pressure during operation mode g (Pa)
$PLR_{adj,n}$	adjustment factor for partial load ratio of the n^{th} chiller
PWEF	present worth escalation factor
Q	cooling load (kW)
$Q_{cap,n}$	cooling capacity of the n^{th} chiller (kW)
$Q_{fan,g}$	fan flow rate during operation mode g (m^3/g)
Q_{load}	cooling load (kW)
Q_{met}	cooling load met by chiller (kW)
Q_{rej}	heat rejection rate (kW)
Q_{wet}	heat transfer rate of completely wet coil (kW)
$\Re$	real number set
r	random number $\in$ U(0,1)
REA	robust evolutionary algorithm formulated in this thesis
run_{max}	number of runs to generate statistically significant results
s	abbreviation of selection for performance graph
S_d	unit duct cost ($/$m^2$)
S_s	set of paths in subsystem s (i.e. supply or return subsystem)
SE	option of selection operator
T	operation time (hr/year)
t	number of epoch
T_0	"initial temperature" at the first generation in simulated annealing
T_a	air temperature (°C)
$T_{adj,n}$	adjustment factor for temperature of the n^{th} chiller

T_{chw}	set point of chilled water supply temperature (°C)
$T_{chw,sp}$	chilled water temperature (°C)
T_{CHWS}	chilled water supply temperature (°C)
T_{cw}	cooling water temperature (°C)
T_{CWR}	condenser water return temperature (°C)
T_{CWS}	condenser water supply temperature (°C)
$T_{dhw,min}$	minimum DHW supply temperature stipulated by the local regulation (°C)
$T_{hw,o}$	hot water outlet temperature of calorifier (°C)
T_{mw}	make-up potable water temperature (°C)
T_s	surface temperature (°C)
$T_{sa,sp}$	set point of supply air temperature (°C)
T_{wb}	wet bulb temperature of cooling air (°C)
TS-R	constraint handling method of tournament selection operator for real-valued GA of Deb (2000)
UA	overall heat conductance (kW K^{-1})
ub	upper bound
V	air flow velocity in duct (m s^{-1})
$\mathbf{v}$	direction vector for particle swarm optimization
V_{cal}	storage capacity of hot water calorifier (m^3)
W	duct width (m)
w	weighting factor for amount of violation for co-evolutionary penalty
$\aleph$	search space of objective function
$\mathbf{x}$	vector of individual with n_{var} number of problem variables
x	abbreviation of recombination for performance graph
$\mathbf{x}^*$	optimal individual of global minimum
$\mathbf{x}_{best,last}$	elite individual of the last run
$\mathbf{x}_{mu1}$	mutated individual by Gaussian deterministic mutation
$\mathbf{x}_{mu2}$	mutated individual by Cauchy deterministic mutation
$\mathbf{x}_{mu3}$	mutated individual by Gaussian stochastic mutation
$\mathbf{x}_{mu4}$	mutated individual by Cauchy stochastic mutation
$\mathbf{x}_{out}$	out-of-bound individual
$\mathbf{x}_p$	vector of parent
$\mathbf{x}_{xo1}$	vector of recombined individual by arithmetic recombination
$\mathbf{x}_{xo2}$	vector of recombined individual by uniform recombination
$\mathbf{x}_{xo3}$	vector of recombined individual by geometrical recombination
XO	option of recombination operator

Greek symbols

α	user-defined parameter for dynamic penalty, parameter for deterministic strategy parameter
β	user-defined parameter for adaptive penalty, parameter for deterministic strategy parameter; or tilt angle of solar collectors (degree)
γ	parameter for deterministic strategy parameter
γ_s	surface azimuth of solar collectors (degree south)
ε_{wet}	effectiveness of completely wet coil
η	stochastic strategy parameter
η_{al}	maximum allowable limit of stochastic strategy parameter
η_f	fan total efficiency
η_m	motor drive efficiency
ι	duct path index
λ	offspring population size in evolution strategy, penalty factor
μ	parent population size in evolution strategy
σ	deterministic strategy parameter
σ_0	initial deterministic strategy parameter
τ	second learning rate for stochastic strategy parameter
τ'	first learning rate for stochastic strategy parameter
ψ_g	fraction of time system operates in mode g

CHAPTER 1 INTRODUCTION

1.1 Background

Owing to the complex nature of a centralized heating, ventilating and air conditioning (HVAC) system, it is increasingly popular to adopt a simulation approach to tackle the related design and management problems, in parallel with the use of practical experience. With the availability of more detailed component models, interaction relationships, and user-friendliness of the development platform, more and more engineering professionals make use of simulation as a tool to analyze the feasibility and performance of a particular design. It is useful to review the changing performance of conventional systems under different climatic and indoor conditions. A simulation model can become a "what-if" evaluator to study possible scenarios. This would be relatively convenient if only one single design variable is involved, and typically parametric study or regression analysis would be applied in this case. However if the problem is related to the interaction between more than two design variables under constrained situations, it would be a challenge to devise a suitable scheme of study. In order to save computational demand and user effort, it is common for personal judgment or intuition to be used in order to facilitate the search progress. The reliability of the "optimal" combination of problem variables may be in doubt, and the solution may be only true within a local region, but not globally true if the search landscape is rugged and multimodal. Such a search process in this context would be normally viewed as optimization, and the goal is to acquire the global or near optimal solution of the problem.

For the in-depth study of HVAC systems, a simulation-optimization approach has been increasingly applied. In the case of optimization problems, a variety of objective functions have been formulated, such as equipment sizing by Wright (1986), control

strategies by Kintner-Meyer (1994), thermal comfort by Huh (1995), plant scheduling by Taylor (1996), routing and distribution by Fong *et al.* (2001), supervisory control by Hanby *et al.* (2002), fault diagnosis by Wang and Wang (2002), energy management by Fong *et al.* (2003, 2004). Particularly in the field of HVAC services engineering, life-cycle cost analysis has increasingly been used by building developers, architects and engineers, as this can help them to look beyond the initial investment and installation costs, and to incorporate the running and maintenance costs properly throughout the life of the systems. The simulation-optimization approach has been also adopted for the decision making process, as demonstrated in the works of Dasgupta (1997), Wright *et al.* (2002) and Nassif *et al.* (2004), as well as the viewpoint of building developers (Kennett 2001). Usually multi-criterion optimization is used, based on the posteriori preference articulation approach, so that a number of possible scenarios are provided to designers and decision makers.

Optimization problems in HVAC services design and operation often have discrete, non-linear, multidimensional and highly constrained characteristics in the search space. In recent years, there have been growing applications of the population-search evolutionary algorithm (EA) in handling different optimization problems. Giraud-Moreau and Lafon (2002) have highlighted that EAs are suitable to handle complex mechanical design problems since no derivative information is required. For instance, Simpson *et al.* (1994) applied genetic algorithm (GA), one of the paradigms of EA, to the optimization of pipe networks. Wright (1996) used GA for the HVAC optimization studies in sizing of HVAC equipment, and Huang and Lam (1997) used GA for optimizing controller performance in HVAC systems. Sakamoto *et al.* (1999) examined the application of GA to optimize the operation schedule for a district heating

and cooling plant, and Asiedu *et al.* (2000) focused on the application of GA for the duct system design. Chow *et al.* (2002) used GA to develop an optimal control scheme for an absorption chiller, and Wright *et al.* (2002) applied a multi-objective GA to identify the optimal pay-off characteristic between the building energy cost and the thermal discomfort of occupants. Angelov *et al.* (2003) applied an EA to propose novel secondary HVAC systems. Lu *et al.* (2004, 2005) adopted GA to optimize the plant operation of a centralized HVAC model.

For the combined simulation-optimization approach, the efficiency of the optimization method for the HVAC simulation models is a primary concern, since the bottleneck for the process is commonly the simulation run for generating the required evaluation function value from the problem variables. Excepting analytical optimization approaches, the working efficiency of any numerical optimization methods is directly associated to the number of evaluation function calls. As a result, a robust optimization method should be able to generate global or near-optimal solutions of HVAC problems with minimum calls to the simulation models. Although a number of the aforementioned research works were based on GA, the working efficiency was inevitably compromised since the population size was commonly in tens or hundreds, leading to a very large number of evaluation function calls to the simulation.

1.2 Appraisal of HVAC Optimization Problems

1.2.1 Formulation of HVAC optimization problems

1.2.1.1 Objective functions

For the HVAC optimization problems, the objective functions are directly related to the expected outcomes of design, operation or both. This may be a maximization or

minimization problem, and it depends on the focus of the optimization under study. Objective functions may cover the following areas:

- maximization of energy saving;
- maximization of efficiency of system or equipment;
- minimization of energy consumption (e.g. net energy, primary energy);
- minimization of cost (e.g. capital cost, operating cost, life-cycle cost);
- minimization of payback period;
- minimization of material;
- minimization of spatial requirement;
- maximization of thermal comfort of indoor environment (e.g. minimization of predicted percentage dissatisfied).

In view of the variety of objectives for optimization, the field can be broadly divided into single objective optimization and multi-objective optimization. For single objective optimization, usually one or more aligned objectives can be combined. In the case of multi-objective optimization, two or more conflicting objectives would be encountered. The expected outcome of the single objective optimization is different from the multi-objective case. The former would give a direct and single optimal or near-optimal solution, while the latter would provide a number of feasible solutions for decision making purpose.

1.2.1.2 Problem variables

The problem variables of the steady state HVAC optimization problems are related to the operating point, mode of operation, component capacity, number of components, stage of operation, set point or physical size. The nature of these problem

variables can be either continuous or discrete. For a typical centralized HVAC system, it broadly includes the air side system and water side system (or secondary system and primary system respectively). From the viewpoints of both design and operation of these sub-systems, the following parameters of the operating fluid would be generally considered to be continuous problem variables:

- mass flow rate;
- supply temperature;
- return temperature;
- supply enthalpy;
- return enthalpy;
- operating pressure.

There are also discrete problem variables for different HVAC sub-systems, such as:

- the number of operating equipment of multiple installations (e.g. multiple chillers);
- the number of steps for staging operation (e.g. step control of reciprocating compressor); and
- the combination of different capacity of operating equipment (e.g. operation with one larger chiller and one smaller chiller in part load requirement).

If the optimization is specifically related to the design problems, the problem variables would be extended to the following aspects:

- capacity of equipment (e.g. capacity of hot water calorifier);
- number of components for heat exchange purpose (e.g. number of rows of cooling/heating coil);
- size of driving component (e.g. impeller diameter of pump or fan);
- dimension of distribution routing (e.g. pipe or duct).

Usually these are discrete problem variables with step increments, since they would be

governed by the available commercial sizes or physical ranges.

1.2.1.3 Constraint functions

There are basically two types: inequality and equality constraint functions. The former are commonly the feasible ranges of the problem variables, while the latter are usually related to the heat or mass balance of different HVAC sub-systems or equipment.

The inequality constraint functions may further include:

- the velocity range of the operating fluid;
- the range of permitted static head loss; and
- the configuration for connection or installation (e.g. succeeding duct size smaller or equal to preceding duct size).

The equality constraint functions in HVAC systems may originate from the following physical law of the sub-systems or the entire system:

- heat balance equation (e.g. relationship between cooling load satisfied and temperature difference of chilled water);
- mass balance equation (e.g. mass flow at mixing tee);
- steady flow energy equation (e.g. vapour compression refrigeration cycle);
- empirical mathematical expression of engineering performance (e.g. pump or fan power at variable flow);
- specific configuration for connection or installation (e.g. size of first duct section equal to fan outlet).

1.2.1.4 Optimization in context

Since most of the HVAC optimization problems of design and energy

management are related to minimization, the development of this research would be based on the minimization problem, which can be described in Eq (1.1) as follows.

$$\underset{x \in \mathfrak{I}}{\text{minimize}}\, f(\mathbf{x}) \equiv f(\mathbf{x}^*) \qquad (1.1)$$

such that $f(\mathbf{x}^*) < f(\mathbf{x})$ for all $\mathbf{x} \in \mathfrak{I} \subseteq \aleph \subseteq \Re^{n_{var}}$

subject to the constraints of

$G_i(\mathbf{x}) \geq 0 \qquad i = 1, \ldots, m$

and $H_j(\mathbf{x}) = 0 \qquad j = (m + 1), \ldots, n_{con}$

where,

$\mathbf{x}$: individual with n_{var} number of problem variables

$f(\mathbf{x})$: objective function value of individual $\mathbf{x}$

$\mathbf{x}^*$: optimal individual of global minimum

$f(\mathbf{x}^*)$: global minimum

G: inequality constraint function

H: equality constraint function

$\mathfrak{I}$: feasible region for constrained problem

$\aleph$: search space of objective function

$\Re$: real number set

n_{var}: number of problem variables

n_{con}: number of constraint functions, $n_{con} \geq 0$

m: number of inequality constraint functions, $0 \leq m \leq n_{con}$

For unconstrained problems, there are no inequality and equality constraints, therefore $\mathfrak{I} \equiv \aleph$.

On the other hand, without loss of generality, the maximization problem can be also handled on a minimization platform with suitable transformation by Eq (1.2) below.

$$\text{maximize } f(\mathbf{x}) = -\{\text{minimize } [-f(\mathbf{x})]\} \qquad (1.2)$$

Usually for an equality constraint function $H(\mathbf{x})$, a sufficiently small positive number δ (e.g. 10^{-3}) would be introduced so that the equality constraint function can be

transformed into an inequality constraint function as follows:

$$\delta - |H(\mathbf{x})| \geq 0 \tag{1.3}$$

In this way all the constraint functions have the same form, and they can be handled together more conveniently throughout the optimization process.

Sometimes the multi-objective optimization can be represented by a single objective approach (Wright *et al.* 2002). This can be implemented by identifying the essential objective function among the available objective functions, and setting the conflicting objective functions into a suitable form of constraint functions.

1.2.2 Characteristics of simulation models in HVAC problems

1.2.2.1 Energy simulation models

In recent decades, different HVAC simulation modeling programs have emerged in both the academic and commercial fields. The typical stages of energy simulation modeling of HVAC problems are shown in Fig. 1.1.

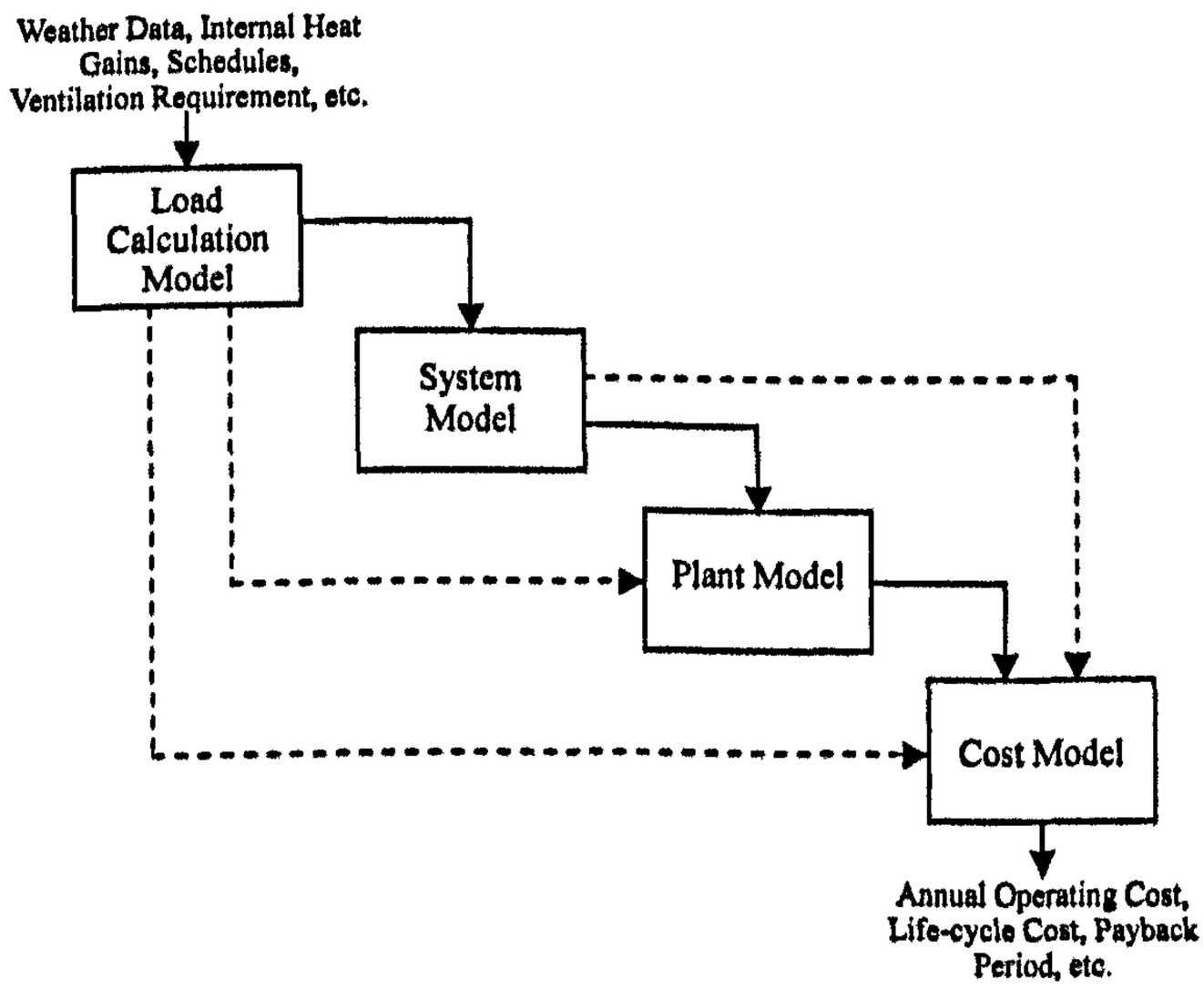


Fig. 1.1. Simulation modeling of HVAC problem

Energy simulation software tools usually adopt the approach of forward modeling, in which the output variables can be predicted by the specified input variables on a platform with the required parameters and established structures. Based on the load-system-plant-cost sequence of an overall HVAC modeling in Fig. 1.1, there are particularly close interrelationship of the outputs and inputs among the load, system and plant models. The algorithm development of the energy simulation programs, such as EnergyPlus (EnergyPlus 2006), DOE-2 (LBL and LANL 1982) and BLAST (BLAST 1993) have been designed with this approach. The heat balance method and weighting-factor method are commonly used to determine the instantaneous space cooling load from the weather condition, building mass and heat gains. In more recent development, the thermal-network method (Sowell 1990) has offered greater flexibility to the users in model building. But this method would demand more computational cost if the building model is rigorously simulated. If the thermal-network approach can be developed in a simplified approach, like the model with lumped parameter of resistance/capacitance network (Ren and Wright 1998, Hanby and Angelov 2000), the model equations can be solved more efficiently.

1.2.2.2 Component-based simulation models

TRNSYS (SEL 2000), IDA Indoor Climate and Energy (Bris Data AB 1999), ESP-r (Hand 2006) and HVACSIM+ (Clarke 1985) are examples of popular component-based HVAC plant simulation programs. In order to provide wide applications and flexible studies of different HVAC equipment and systems, these programs are based on a library of component models. These are mostly steady state models, although the provision of dynamic simulation is included in these plant

simulation programs. The components of equipment cover those from both the air side and water side of a centralized HVAC system. The components of air side system mainly cover cooling and dehumidification coils, heating coils, humidifiers, fans, dampers and ducts. For the components of water side system, there are chillers, boilers, cooling towers, thermal storage equipment, cogeneration plant, etc.

For the mathematical modeling of cooling and dehumidification coil, it is a heat and mass transfer component since both sensible heat and latent heat are involved. For instance in TRNSYS 15.3 (SEL 2000), the component model of cooling and dehumidification coil adopts the air effectiveness from Braun (1988) and the efficiencies of rectangular-plate fins from Threlkeld (1970). In this mathematical model, the effectiveness of the wet coil would depend on the outlet chilled water temperature, so iterative method is required to find the outlet states of the conditioned air.

For the mathematical modeling of a fan, the power is commonly determined as normalized power, which is described by a polynomial expression with suitable order of the normalized mass flow rate. The coefficients of the polynomial would be based on the manufacturer's information. A similar approach would also be applied to pump modeling.

Owing to the relatively complex nature of the central plant and water side system, it is a challenge to develop the mathematical modeling in detail. Therefore these components are also commonly modeled with suitable mathematical regression expressions, which cover both full- and part-load operation. These mathematical expressions are commonly derived in the format of exponentials, polynomials or Fourier series, but the second or third order polynomial approximation is the most common (Yik and Burnett 1998, Yik and Chan 1998). If two variables are involved, it is commonly

represented by a biquadratic expression. In the available commercial plant simulation programs, the pressure balance of the air or water distribution system is usually not considered. If this level of detail is required, additional customized components should be developed.

1.3 Research Objectives

The aims and objectives are:

a. to advance the use of numerical and heuristic optimization method in the design and energy management of HVAC systems;

b. to devise an effective coupling linkage between an optimization algorithm and the simulation model, which would be implemented by using a common plant simulation program; and

c. to develop a robust optimization platform, which can solve single objective minimization problems at a reduced number of simulation function calls.

1.4 Research Methodology

In order to achieve the objectives in this research work, the following methodology has been adopted:

a. To conduct the literature survey of the optimization and simulation methods, with the focus on the problems of HVAC engineering design and energy management.

b. To establish plant simulation models of typical centralized HVAC systems in Hong Kong by using suitable component-based modeling program.

c. To build up a prototype optimization suite in the context of EA, and develop an appropriate coupling link to the HVAC plant simulation models for optimization

purpose.

d. To formulate the essential constituents and operators of a robust EA (REA) through experimentation with a variety of standard test functions and test problems.

e. To benchmark the performance of the REA against other optimization methods in the field of EA.

f. To verify the effectiveness of the REA through example HVAC problems in the areas of design and energy management, which reflect the potential local applications in Hong Kong.

In order to carry out the work in this research, the following facilities and software have been applied:

a. contemporary personal computers (configuration of Pentium 4, 1 GB RAM, 60 GB hard disk with flexible virtual memory, operating system of Windows XP);

b. plant simulation program TRNSYS 15.3 (plant simulation engine) and IISiBat 3 (graphical input/output interfacing program for TRNSYS);

c. Visual FORTRAN 6.5 (for compiling new components for TRNSYS, and recompiling the executable TRNSYS simulation engine); and

d. MATLAB 6.1 (for developing the EA Suite and coupling linkage);

e. Microsoft Excel (for sensitivity test, 2D and 3D graph plotting, and graphical analysis).

f. Microsoft Word (for thesis writing and editing).

1.5 Organization of Thesis

There are ten chapters in this thesis. Chapter 1 is the introductory part, to show the background of simulation-optimization approach and to appraise the characteristics of HVAC optimization problems. Chapter 2 reviews the traditional and heuristic optimization methods, with focus on the effectiveness of EA and popular constraint handling techniques. Chapter 3 presents the development of a component-based HVAC system model by using the simulation program TRNSYS. Chapter 4 discusses the development of EA Suite in MATLAB and its coupling linkage to the HVAC system model. The EA operators including mutation, selection, recombination, constraint handling, as well as the supplementary operation strategies are described in Chapter 5. Chapter 6 reports the setup of empirical studies across different combinations of EA operators through a variety of test functions. Chapter 7 formulates the best combination of EA operators from the results of full empirical runs, and proposes the REA for HVAC optimization problems. The qualitative analysis of constituents of the REA, and benchmarking with some standard methods in EA field are described in Chapter 8. Chapter 9 verifies the performance of the REA in solving the example practical HVAC optimization problems in the areas of design and energy management. Chapter 10 summaries the research outcomes and suggests future work. Details of test functions / problems for experimentation and qualitative analysis are consolidated in the appendix.

CHAPTER 2 OPTIMIZATION APPROACHES FOR HVAC PROBLEMS

After understanding the features and possible applications of HVAC optimization problems in Chapter 1, the common optimization approaches already applied for the HVAC problems are reviewed and described in this chapter. Both traditional and heuristic optimization methods are discussed. Since the focus of this research is the suitability of EA in handling the HVAC optimization problems, the characteristics and discrepancies of different paradigms of EA are analyzed in detail. The advantages of EA over the traditional and other heuristic optimization methods in handling HVAC problems are discussed and highlighted. In addition, constraint handling techniques are needed to tackle constrained HVAC problems, so the popular methods are also reviewed and presented in this chapter.

2.1 Common Optimization Methods for HVAC Problems

A number of optimization methods have been applied in handling HVAC optimization problems with respect to their multidimensional, nonlinear, mixed continuous-discrete and highly constrained nature. There has been continuous development of optimization methods in last two decades and such problems can be effectively tackled according to their nature and the optimal solution identified. These methods include the analytical approach (Söylemez 2001), gradient-based and nonlinear programming methods (Zheng and Zaheer-Uddin 1996, Hanby and Angelov 2000, Zaheer-Uddin and Zheng 2000), direct search methods (Wright and Hanby 1987, Hanby and Wright 1989, Lambert and Wright 1991), dynamic programming (Rink and Li 1995, Fong *et al.* 2001), artificial neural network (Yang *et al.* 2003). These methods all have their roles in certain optimization problems, and they have their own features as well as

limitations in application. The methods are briefly reviewed here to provide a context for the research and to serve as a background for the development of the search algorithms in this thesis.

2.1.1 Analytical approach

Differentiation is a classical optimization method to determine the optimum for a continuous objective function. This is efficient for a smooth function with a single problem variable. Once the number of problem variables is up to two or more, the roots from the partial derivatives produce families of lines, and the optima can be determined from the intersection of these lines. The global optimum can be identified from a group of optima in an effective way. If equality constraint functions are involved, the method of Lagrange multipliers can be applied to include the constraint functions into the objective function (Sanaye and Malekmohammadi 2004). The roots of the partial derivatives of this modified objective function still produce an equation set to determine the optimum. To handle inequality constraints, usually the Kuhn-Tucker (KT) method would be adopted. However the solution of the KT equations is just the necessary condition for optimality (Gouda *et al.* 2002), so the involvement of constraint functions may impede the search reliability of optimum. The obvious limitation of the analytical approach is its incapability to handle optimization problems with discrete variables, or those without first and second derivatives. It also has problems with multimodal optimization since it may be easily trapped at a local optimum.

2.1.2 Gradient-based and nonlinear programming methods

The method of steepest descent and the Newton method are classical

gradient-based numerical methods, and they have been in use for more than half a century. The method of steepest descent starts at an arbitrary point on the search space and searches along the direction of the steepest gradient. The new gradient is orthogonal to the previous one, so that the search is gradually directed towards the optimum. As compared to the steepest descent method, Newton method has a significant improvement of the searching efficiency. This is achieved by working on an approximation from multidimensional Taylor series expansion of the objective function and the corresponding matrix of second derivatives called Hessian (Jacobian) matrix. However the Hessian matrix of Newton method may be difficult to determine, and the search of optimum may be unstable and possibly erroneous in multimodal problems. As a result, some quasi-Newton techniques have been developed to replace the Hessian matrix by an approximation one, but necessary checking and remedial works are included in order to provide a trustworthy solution. Two popular quasi-Newton methods are DFP (Davidon-Fletcher-Powell) algorithm (Powell 1964) and BFGS (Broyden-Fletcher-Goldfarb-Shanno) algorithm (Broyden 1965; Fletcher 1963; Goldfarb and Lapidus 1968; Shanno 1970). The Newton and quasi-Newton methods afford rapid convergence and search efficiency.

The sequential quadratic programming (SQP) algorithm is a generalization of the Newton method for constrained optimization. It would find a step away from the current point by minimizing a quadratic model of the problem. Every iteration a quadratic approximation is developed for the Hessian matrix of the Lagrangian function. This is then used to generate a QP subproblem whose solution is used to form a search direction. The trust-region SQP is useful to handle the constrained problems with linear equalities (Furey 1993, Dong *et al.* 2003), and this method provides a neighbourhood for the current

iterate to determine a better search direction. The active set SQP is applied for general nonlinear optimization including both linear and nonlinear constraints (Zheng and Zaheer-Uddin 1996, van Schijndel 2002, Liu 2005). Its computational efficiency is enhanced by approximating the problem with the active set of constraints. The inactive constraints are omitted.

In practice, the GRG2 method (Lasdon *et al.* 1978) has been used to handle constrained optimization problems with numerous problem variables and linear equality constraints, and it was applied to optimization problems of HVAC systems (Silverman *et al.* 1981; Murray 1984). Implementation of GRG2 is based on GRG (generalized reduced gradient) algorithm. Firstly the equality constraints are used to eliminate a subset of the problem variables, so that the original problem can be changed to a bound-constrained problem of the remaining variables. GRG2 is commonly initialized with a feasible solution and retains feasibility of solution along the optimization process. The BFGS quasi-Newton method is adopted to determine the search direction. The approximated Hessian matrix is handled as a dense matrix, so the involved active constraints can be manipulated more effectively.

However these gradient-based methods, similar to those of the analytical approach, are limited to continuous and unimodal functions with the existence of the first and second derivatives.

2.1.3 Exhaustive search

The exhaustive search method does not need any derivative information of objective and constraint functions, but it demands a considerable computational time, particularly if the search interval for each variable is divided too small. If two problem

variables are involved, a coarse topographical landscape can be developed first. Then the possible location of the global optimum can be identified, and the exhaustive search can be focused on the potential region with fine resolution. However if the problem is multidimensional and has more than two problem variables, the topographical preview cannot be implemented. On the other hand, the degree of resolution may be based on the optimization experience or the bounds of the problem variables. If the resolution is still relatively coarse with respect to the problem, the topographical landscape may even mislead the location of the global optimum. For instance, the topography of a multimodal search landscape may be misunderstood as a unimodal one if the search points are under-sampled. Of course, these worries can be minimized by magnifying the resolution as large as possible, but this would in turn exponentially increase the computational cost.

2.1.4 Direct search

The direct search is another method that information about derivatives is not required. These have the attractive property of working with a population of points as well. This is a significant improvement over the exhaustive search in terms of optimization efficiency. The development of the direct search method is exemplified by the pattern search algorithm of Hooke and Jeeves (1960). The search starts from an initial feasible point in the search space. A series of exploratory probes that form a searching mesh are made along the axes of problem variables. The objective function value of each probe is evaluated, recorded and compared, so that it can inform an accelerated move to a new base point for another set of exploratory probes. This process continues until the optimum is determined. Another common direct search method is

the complex method (Box 1965, Nelder and Mead 1965). Its principle is to explore the optimum by rejecting infeasible points. Firstly a group of trial points, called simplex, are generated randomly within the feasible space. All the points in the simplex are checked for their feasibility against the constraint functions. If any trial point has constraint violation, it would be successively moved halfway back towards the centroid of the existing points of the simplex until it is feasible. Therefore the infeasible point would not be maintained, but relocated within the feasible space.

In order to avoid being trapped at a local optimum in direct search, an improved method of iterative stochastic hillclimbing has been introduced. In this method, successive restarting searches are implemented once there is no further improvement of the solution. Finally the optimum would be identified among the results of these multiple runs. Of course, such an improvement must be balanced against an increase in computational time due to the repetitive search.

The direct search methods have been applied for handling the HVAC optimized design problem (Wright 1986, Wright and Hanby 1987, Lambert and Wright 1991). But these methods have difficulties with constrained problems where the solution lies on the boundary of any constraint functions. Unfortunately, it is quite common for the global optimum to be close to or even touch the constraint functions in HVAC optimization problems. Therefore prior knowledge of the optimization problem is required if direct search methods are used.

2.1.5 Dynamic programming

Dynamic programming is a method for solving an optimization problem by memorizing the sub-problem solutions rather than recomputing them. This is useful to

tackle optimal supervisory control and sequential decision problems (Rink and Li 1995, Lin and Yi 2000, Fong *et al.* 2001). The three main steps of dynamic programming are initialization, matrix-fill and trace-back. In general, dynamic programming has a simple and logical nature. It works well to find the optimum for problems that can be formulated according to the structure of dynamic programming. However not all kinds of HVAC optimization problems can be transformed into the required format of this method. The memory-demanding feature of dynamic programming may be another limitation if the dimension of problem variables and their operating ranges are large. In this case, any approach which can reduce the computational resources should be adopted, such as to apply suitable bounds or constraints to the problem variables for facilitating the search (Chen 2001).

2.1.6 Artificial neural network

Artificial neural network (ANN) is commonly applied in the plant and system modeling if the analytical mathematical descriptions cannot be easily determined. Usually based on the empirical input and output data, the plant model can be trained and formulated at last. ANN itself is seldom a direct optimization tool to handle the common objective functions in the HVAC regimes, however it would produce an optimal plant or system model that can be used to evaluate the problem variables being searched by an external optimizer. This optimization tool should be well articulated with ANN to produce the optimal solution according to the nature of the objective and constraint functions (Gerald and Gibson 1997, Chow *et al.* 2002). Although some literatures would indicate that the developed ANN can be utilized for both learning and optimization purposes (Yang *et al.* 2003), this does not mean the nature of ANN can be used for

optimization. Usually in this situation, a solution set has been generated by certain means of parametric study of the problem variables in advance, and ANN just makes use of this available optimization information during the learning process. Since the solution set is acquired from parametric study, the available optimization information would depend on the degree of resolution and combination, and this information would not be accurate if the search landscape is complicated.

2.2 Heuristic Optimization Methods except EA

2.2.1 Simulated annealing

Simulated annealing, an optimization scheme to prevent from being trapped at the local optimum, was inspired by Kirkpatrick *et al.* (1983) from the annealing phenomenon of hot metals. Koeppel *et al.* (1995) adopted this optimization method to determine a one-week optimal control of an absorption chiller system. The search of simulated annealing is carried out from generation to generation. Each generation there are one parent and one offspring. If the fitness of offspring is better than that of parent, the offspring would replace the parent. If the offspring has worse fitness, it can still become the next parent with certain probability, called annealing probability. However this probability is exponentially decreased as the search progresses, so the chance to retain the less fit offspring would be decreased. For minimization, the annealing probability at the k^{th} generation (where $k \in \mathbf{I}^+$) is determined as follows:

$$p_{a,k} = \exp\left(-\frac{f_o - f_p}{c^{k-1} T_0}\right) \qquad (2.1)$$

where,

$p_{a,k}$ = annealing probability at k^{th} generation

f_o = function value of offspring

f_p = function value of parent

c = "cooling rate", $c \in (0,1)$

T_0 = "initial temperature" at the first generation, i.e. $k = 1$

Since $p_{a,k}$ gets smaller as the search progresses (i.e. k gets larger), there is a progressively lower chance that a worse individual would be accepted. Although simulated annealing has single searching approach, the existence of $p_{a,k}$ would allow the worse individual to be retained, so that the search can escape from the trend dictated by the local optimum. By using this method, the annealing parameters like "initial temperature" and "cooling rate" have to be decided first. However these parameters would be problem-specific, so it is necessary to have certain prior knowledge of the problem domain, or to use a trial-and-error approach to determine them.

2.2.2 Tabu search

Glover (1989, 1990) designed a heuristic method called tabu search that allows the search to escape from a local optimum. The fundamental idea of tabu search is to continuously explore the new solution in the search space by avoiding move reversals to the solution space that has been previously visited. The avoidance of move reversals is implemented through forbidding or penalizing such moves, so it is called "tabu". In this regard, this method records recent moves in one or more tabu lists, which become a search memory in the course of optimization. The influence of the tabu list can be overridden by an aspiration level if the move is sufficiently good, then a reverse move would be permitted. To determine the best move in the current iteration, the assumption is made that good moves are more likely to reach the optimum or near-optimum in the search space. Wen and Chang (1997) applied tabu search for optimal planning of an electric power transmission network. Youssef *et al.* (2001) reported that in the spatial

optimization problem, tabu search can perform more effectively than simulated annealing.

2.2.3 Particle swarm optimization

Particle Swarm Optimization (PSO) was first proposed by Kennedy and Eberhart (1995). This heuristic method was motivated by the social behaviour of organisms such as bird flocking and fish schooling. As an optimization tool, PSO provides a population-based search procedure in which the particles (individuals) change their positions (states) with time. At the start of PSO, the first population is initialized randomly around the multidimensional search space. During the flight, each particle adjusts its position according to its own experience, as well as the experience of neighbouring particles, making use of the best position encountered by itself and its neighbours. At the k^{th} generation (where $k \in \mathbf{I}^+$), the population $\mathbf{x}_k$ is improved to become new population $\mathbf{x}_{k+1}$ as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_{k+1} \tag{2.2}$$

where,

$\mathbf{v}_{k+1}$ = $\mathbf{v}_k + [c_1 r_{1,k} (\mathbf{p}_{best,k} - \mathbf{x}_k) + c_2 r_{2,k} (\mathbf{g}_{best,k} - \mathbf{x}_k)]$,
direction vector generating the new population from the k^{th} one

$\mathbf{p}_{best,k}$ = $\arg\min_{\mathbf{x} \in \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k\}} f(\mathbf{x})$,
location vector of the particle with the best fitness up to the k^{th} epoch

$\mathbf{g}_{best,k}$ = $\arg\min_{\mathbf{x} \in \{\mathbf{x}_{0,0}, \mathbf{x}_{0,1}, \ldots, \mathbf{x}_{0,k}, \ldots, \mathbf{x}_{n,0}, \mathbf{x}_{n,1}, \ldots, \mathbf{x}_{n,k}\}} f(\mathbf{x})$,
location vector of the particle with the best fitness among the entire population up to the k^{th} epoch, n is population size of the particles

r_1, r_2 = random number $\in \mathbf{U}(0,1)$

c_1, c_2 = problem-specific scalars

From Eq (2.2), the new position of population would be determined by the particle with

the best fitness of each individual up to the k^{th} generation $\mathbf{p}_{best,k}$, as well as the particle with the best fitness across the whole population up to the k^{th} generation $\mathbf{g}_{best,k}$. Therefore PSO can be interpreted as a hybrid approach driven by both global and local search. The stochastic nature of PSO provides a flying momentum and allows thorough search around different regions of the multidimensional space, leading the population towards the global optimum.

2.2.4 Ant colony optimization

Ant colony optimization (ACO) is useful to solve the discrete optimization problem, such as the travelling salesman problem. The development of ACO by Dorigo and Maria (1997) was inspired from the behaviour of real ant colonies in the nature. The observation is when the ants choose a shorter path, they would create a trail of pheromone (an odorous chemical from ant) faster and stronger than the ones walking a longer path. Since the stronger pheromone would attract the other ants better, more and more ants would pick the shorter path, until all the ants have found the shortest path. To prevent the premature convergence of the optimization problem due to the excessive virtual pheromone in the ACO process, a measure of "pheromone evaporation" is implemented.

Clearly ACO is a probabilistic population-based search method, but it relies heavily on problem-specific parameters, such as the evaporation constant and a number of weighting factors. The problem variables should be discrete, rather than continuous. Of course a continuous domain for each problem variable can be approximated by a discrete step with the required resolution, but this would add one more problem-dependent parameter to the optimization process.

2.3 Paradigms of EA

EA is a probabilistic and population-based heuristic algorithm developed from the Darwinian paradigm of evolution, which is often viewed as analogous to optimal exploration and optimization. The essential steps are derived from the fundamental principles of *variation* and *selection* of the Darwinian evolution throughout generations. EA can be conceptually presented by the evolutionary loop as shown in Fig. 2.1.

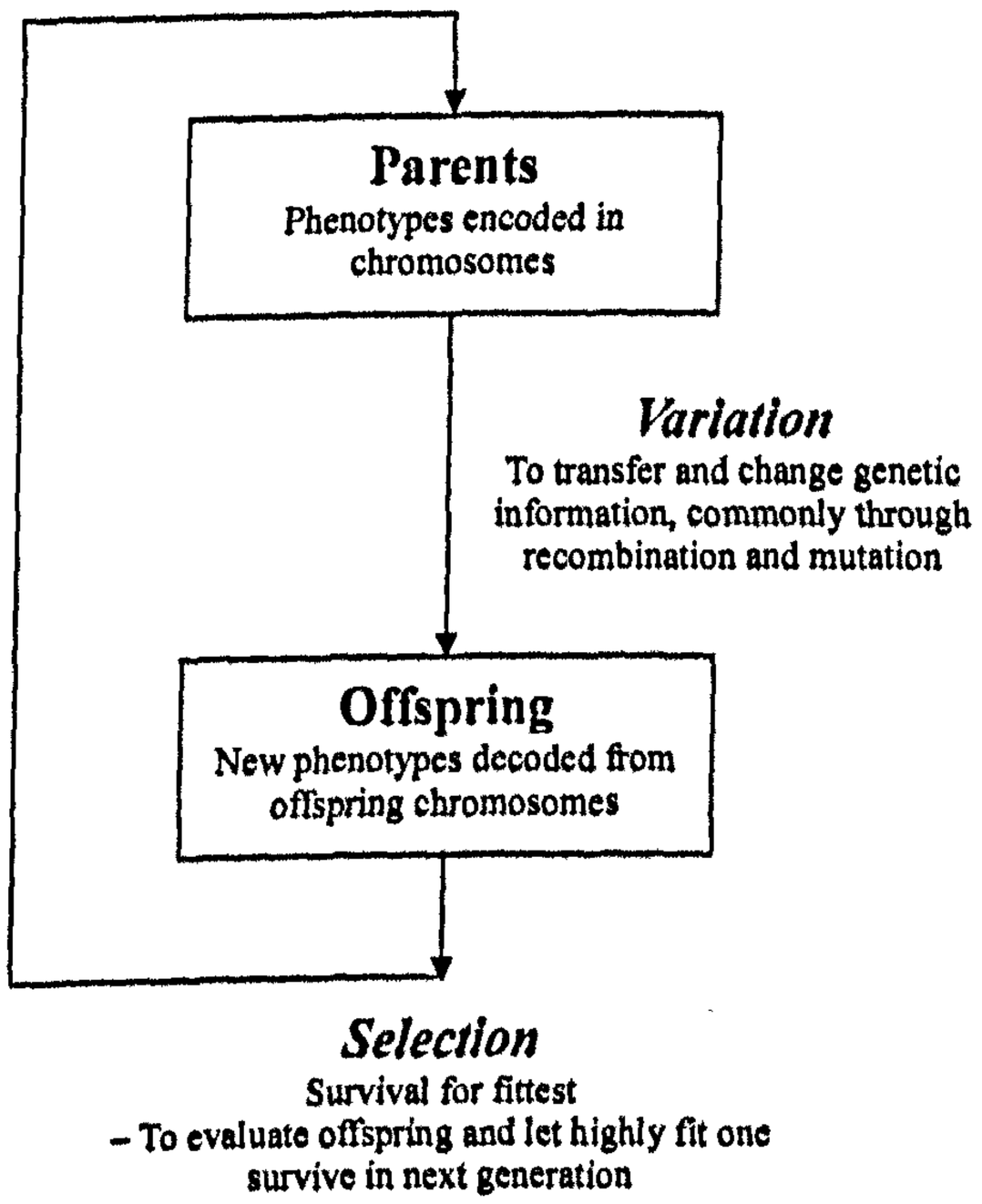


Fig. 2.1. Evolutionary loop

The chromosomes of any individual are represented in genotypes, which define the unique nature of the individual, and the phenotypes of the corresponding genotypes show the traits of the individual. In the other words, the phenotypes are mapped to genotypes, or encoded to become the chromosomes. In the evolutionary loop, the genetic information of the parents would be transferred and changed through the variation process, and then the chromosomes of offspring would have new genotypes

hence new phenotypes. Selection is to implement "survival for fittest". The fitness of offspring is evaluated first, and the highly fit offspring would be selected to survive in next generation. The fitness of an individual refers to the performance of adaptation to the environment for survival according to the Darwinian paradigm of evolution. In typical optimization problem, it is the evaluation outcome of the fitness function, typically the combined effect of objective function and constraint violation incurred. Therefore an individual with higher fitness means greater optimal potential, in turn larger chance for survival. The interrelationship among chromosome, genotype and phenotype is illustrated in Example 2.1.

Example 2.1

Assume an optimization problem with one problem variable. The fitness of individual is simply equal to its value in integer. The population size is two, and each individual is represented in binary string in 5-bit. Assume the details of two parents are as follows. The phenotypic value of each parent is transformed (encoded) into the binary value, i.e. genotype, as well as the corresponding chromosome with the length of five.

	Phenotype	Genotype	Chromosome
Parent A	$2 (= 2^1)$	00010	□□□□□
Parent B	$5 (= 2^2 + 2^0)$	00101	□□□□□

Assume after variation (including recombination and mutation), the offspring becomes two new chromosomes, with the corresponding genotypes and phenotypes as follows:

	Chromosome	Genotype	Phenotype
Offspring A	□□□□□	00110	$6 (= 2^2 + 2^1)$
Offspring B	□□□□□	01001	$9 (= 2^3 + 2^0)$

Since selection for fittest is applied, the *offspring B* would be selected for next generation since its phenotypic value is 9. □

For EA in general, there are three major paradigms – genetic algorithm (Holland 1975, Goldberg 1989), evolutionary programming (Fogel 1964, Fogel 1992), and evolution strategy (Rechenberg 1973, Schwefel 1981). The key genetic operators used in EA are recombination, mutation and selection. Briefly, *recombination* is a reproduction process of parents, *mutation* is to alter genetic information of parents, and *selection* is to choose offspring for survival. The major discrepancies among genetic algorithm, evolutionary programming and evolution strategy can be analysed from their data representation, degree of importance of recombination and mutation, and the approach of selection. The background and characteristics of the three paradigms are introduced in the ensuing sections.

2.3.1 Genetic algorithm

The genetic algorithm (GA) was developed by J.H. Holland in the 1960s (Holland 1962, 1967). On the other hand, H.J. Bremermann offered conceptually equivalent procedures also in the 1960s (Bremermann 1962, Bremermann *et al.* 1965), and A.S. Fraser even earlier in 1950s (Fraser 1957). GA follows closely to the paradigm of Darwinian biological evolution, so it has an emphasis on crossover (recombination) and a probabilistic selection operator. Mutation plays a minor role and it is treated as a background operator. Binary strings are commonly used in GA for representation of problem variables under optimization, although real-valued representation can be also used. There is a GA-related paradigm called genetic programming, which makes use of the GA operators to optimize the objective of algorithmic program, instead of mathematical function. Holland (1975) has introduced the simple genetic algorithm (SGA) with the typical procedures shown in Fig. 2.2.

```
epoch = 1;
initialize the first population;
evaluate fitness of the population;
for epoch = 2:epoch_max do
        encode the population;
        select parents for crossover;
        crossover of parents with p_c;
        mutate individuals with p_m;
        decode offspring;
        evaluate fitness of offspring;
od
```

Fig. 2.2. Pseudo-code of GA

In the first epoch, a population is initialized from the search space. From the second epoch, each individual originally in integer or real number (phenotype) is encoded in binary strings (genotype) and form a chromosome according to the required precision or bit size. Two parents are generally chosen by proportional selection, which is based on relative fitness of the parents in the population. The individual with the higher fitness will have greater chance of selection. Proportional selection is commonly implemented by using a roulette wheel as shown in Fig. 2.3. Crossover of these two parents would be implemented with the crossover probability p_c. Among different crossover methods, the single-point crossover is popular and straightforward. For mutation, single bit-flip mutation method is common. Since mutation is not emphasized, so the individual after crossover would undergo the mutation process with low mutation probability p_m, usually equal to $\frac{1}{L}$, where L is chromosome length. Then the individual would be decoded from binary string to integer or real number form for evaluation purpose. Example 2.2 demonstrates the crossover and mutation processes by using the two parents already mentioned in Example 2.1.

Goldberg (1989) has described wide applications of GA in handling optimization problems in the fields of science, engineering and business. GA can promote satisfactory searching performance from the building block hypothesis and the schema theorem. This has led to GA becoming popular in the field of heuristic and engineering optimization in recent decades.

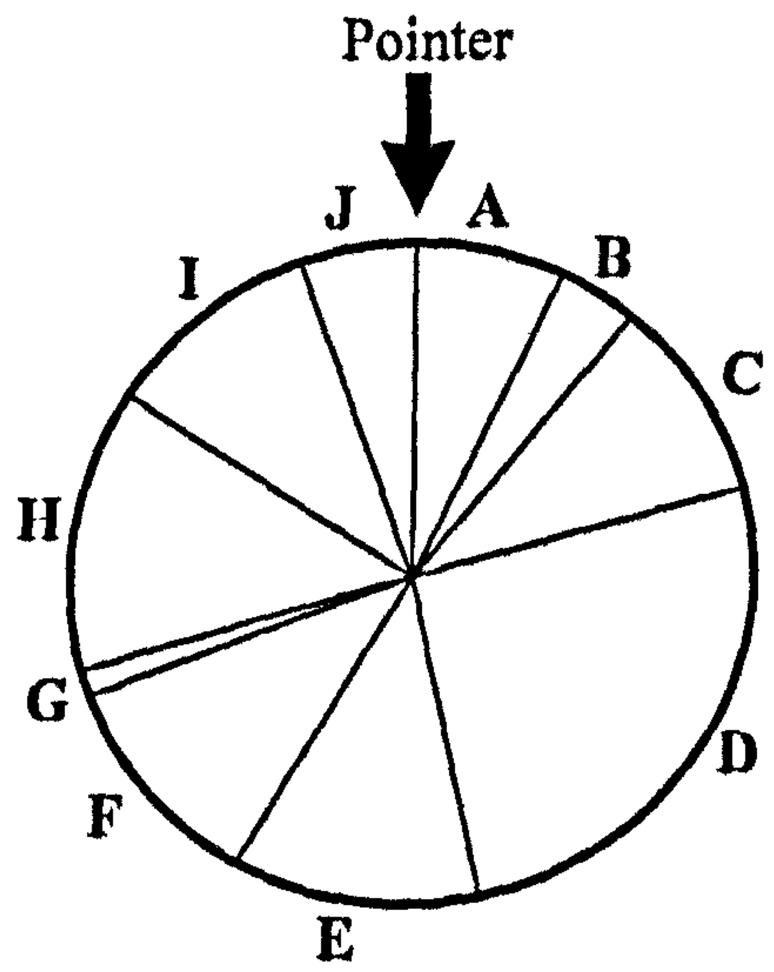
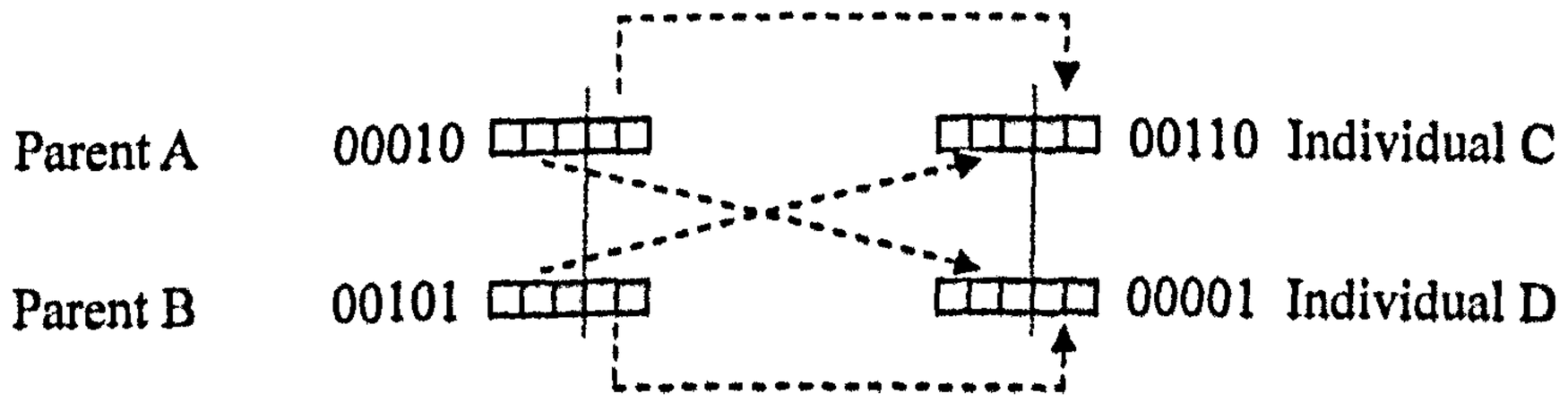


Fig. 2.3. Roulette wheel. It contains the sectors of a population with 10 individuals. The area of each sector is proportional to the fitness of the corresponding individual. Therefore in every spin of proportional selection, Individual D would have the greatest chance indicated by the pointer thus selected, while Individual G the least.

Example 2.2

Follow the information of the two parents in Example 2.1. Assume crossover is allowed and a single-point crossover is implemented after the third bit, the offspring becomes:



For mutation, assume individual C remains intact but individual D is probabilistically allowed for mutation. Assume that the second bit of individual D is chosen randomly and a bit-flip is implemented. Therefore the two offspring becomes as follows:

	Chromosome	Genotype
Offspring A	☐☐☐☐☐	00110 (same as Individual C, no change)
Offspring B	☐☐☐☐☐	01001 (second bit flipped from 0 to 1 in Individual D)

2.3.2 Evolutionary programming

Evolutionary programming was developed by L.J. Fogel in 1960s (Fogel 1962, 1964), and further refined by D.B. Fogel in the 1980s and 1990s (Fogel 1988, 1992, Fogel and Fogel 1988). The original motivation for developing evolutionary programming was to generate finite-state machines for solving time-series prediction tasks. Afterwards, evolutionary programming was also adopted in parameter optimization problems. The framework of evolutionary programming is shown in Fig. 2.4.

```
epoch = 1;
initialize the first population;
evaluate fitness of the population;
for epoch = 2:epoch_max do
        mutate parents to become offspring;
        evaluate fitness of offspring;
        select parents for next generation;
od
```

Fig. 2.4. Pseudo-code of evolutionary programming

Evolutionary programming has the emphasis on mutation operator instead of recombination operator, so recombination is not common in this paradigm. Usually each individual contains a problem variable and corresponding strategy parameter, where the latter is used to provide search step and direction for the former (i.e. the problem variable). This pair would be evolved with Gaussian distribution throughout the

generations in evolutionary programming. Since the selected highly fit offspring should result from a robust strategy parameter for optimal search, it shows that the strategy parameter is also self-adapted along the evolution of the problem variable. In evolutionary programming, stochastic tournament selection is commonly used, the parents for next generation are selected from the winners of the competition against a specified number of opponents, which are stochastically selected from the united group of parents and offspring.

2.3.3 Evolution strategy

Evolution strategy was developed by I. Rechenberg and H. Schwefel in 1960s (Rechenberg 1965, Schwefel 1965). Evolution strategy is commonly used in problems with real-valued or discrete variables. The genetic operators of evolution strategy include recombination, mutation and selection. Evolution strategy emphasizes the equal importance of mutation and recombination. A recombination operator has not been proven to provide a definite performance gain, but it would probably reduce the occurrence of scattered individuals as compared to the effect due to mutation operator. For the mutation operator, similar to evolutionary programming, the Gaussian realization with self-adapted strategy parameter is used. The framework of evolution strategy is presented in Fig. 2.5.

```
epoch = 1;
initialize the first population;
evaluate fitness of the population;
for epoch = 2:epoch_max do
        recombine parents;
        mutate the recombined individuals to become offspring;
        evaluate fitness of offspring;
        select parents for next generation;
od
```

Fig. 2.5. Pseudo-code of evolution strategy

Generally evolution strategy would have a parent population size μ and an offspring population size λ. The implementation of evolution strategy can be broadly categorized into the commas strategy (μ, λ) and the plus strategy $(\mu + \lambda)$, and this classification is based on the selection approach to be adopted. For the (μ, λ) strategy, it is common that $\lambda = k\,\mu$, where $\lambda > \mu$ and $k \in I^+$. The λ offspring would become the selection pool and they are all ranked according to their fitness. Then the best μ individuals would be deterministically selected to be the parent population for next epoch. For the $(\mu + \lambda)$ strategy, the selection pool is the union of μ parents and λ offspring, all the individuals are ranked according to their fitness, and the best μ individuals would be selected to be the next parent population deterministically.

2.3.4 Summary of major features of EA paradigms

For the operators and characteristics of the three major paradigms of EA, their feature, similarity and difference are summarized in Table 2.1.

Table 2.1. Summary of major operators of GA, evolutionary programming and evolution strategy

	Genetic Algorithm	Evolutionary Programming	Evolution Strategy
Recombination (crossover)	Core operator, with recombination probability > 0.5.	Not included.	Important operator, with recombination probability = 1.
Mutation	Background operator, with low mutation probability, usually 1/L (where L is chromosome length).	Core operator, commonly using stochastic strategy parameter with Gaussian realization, with mutation probability = 1.	Core operator, commonly using stochastic strategy parameter with Gaussian realization, with mutation probability = 1. Usually mutation follows recombination.
Selection	Probabilistic selection of parents for recombination and mutation (selection for reproduction).	Probabilistic selection of offspring to be parents of next epoch (selection for survival).	Deterministic selection of offspring to be parents of next epoch (selection for survival).

The paradigms of evolutionary programming and evolution strategy are similar, except there is no recombination in the former and deterministic selection instead of probabilistic selection in the latter. On the other hand, there are several significant features of evolution strategy and evolutionary programming compared to GA:

a. For GA, recombination is the core operator, mutation is much less significant. However mutation is the core operator for both evolutionary programming and evolution strategy.

b. In GA, the recombination and mutation probabilities are commonly less than one. This means that some individuals within the population would remain intact without evolution. But for evolution strategy, both probabilities are one. This implies that these two operators would be definitely involved for variation of individuals. For evolutionary programming, the probability of mutation is also commonly one.

c. Apart from the nature of selection, the approach and sequence of selection in their evolution loops are different. For GA, the parents would be selected for the ongoing

recombination and mutation processes through the selection operator, selection is therefore used for those variation operators and implemented before the variation process. But for evolutionary programming and evolution strategy, selection is applied on the offspring population to generate the parent population of next epoch, so selection is implemented after the variation operators. In summary, the approach of "selection for reproduction" is adopted in GA, while "selection for survival" in both evolutionary programming and evolution strategy.

As a result, although there are similarities among the three paradigms, there are distinguishing features for each paradigm, so their optimization performance would be different. The primary objective of this research is to explore alternative configurations and varied combinations of EA operators, so that the robustness of the optimization algorithm is not limited by the traditional format of the respective EA paradigms. In particular, emphasis is placed on the effectiveness in terms of the number of function calls in the evaluation phase, as this is the rate-determining step in solving many HVAC optimization problems.

In the context of HVAC optimization problems, a robust EA should have the following performance characteristics:

- Exploitative search instead of blind search;
- Exploring optimum even at the rugged search landscape;
- Minimum but sufficient evaluation function calls;
- Good operational connection among different EA operators and evaluation function.

With deeper understanding of the epitasis and modality of the optimization problems, i.e. the loci of search and the landscape of search space, the effectiveness of different involvement and combination of EA operators is examined systematically in this thesis.

2.4 Advantages of EA over Different Optimization Methods

According to the no free lunch theorems advocated by Wolpert and Macready (1997), there is no single optimization method which can solve all kinds of problems and give the best performance. In the context of HVAC optimization problems which are developed using a detailed component-based model, the complex interrelationship and possibly discrete nature of the problem variables means that these problems cannot be solved by the traditional analytical or gradient-based methods. Although there are continuously emerging new numerical or heuristic optimization methods, their appropriateness for the HVAC problems should be reviewed. The features and advantages of EA against the deficiencies of common optimization methods as described in Sections 2.1 and 2.2 are summarized in Tables 2.2 and 2.3 respectively. From this review, EA is advantageous to common optimization methods, as well as other heuristic optimization methods.

As mentioned in Section 1.1, most of the HVAC optimization problems reported in the literatures were based on the paradigm of GA (Simpson *et al.* 1994, Wright 1996, Huang and Lam 1997, Sakamoto *et al.* 1999, Asiedu *et al.* 2000, Chow *et al.* 2002, Angelov *et al.* 2003, Lu *et al.* 2004, 2005). The unique features of GA are its binary representation of problem variables, and the significant role of recombination for offspring generation. Although a number of optimization problems have been effectively handled by GA, the performance and efficiency of other paradigms – evolution strategy and evolutionary programming – has been seldom reported. Although Giraud-Moreau and Lafon (2002) suggested that evolution strategy would have better performance over GA in handling constrained mechanical design problems, the comparison was only based on the elementary recombination and selection operators, and

there was no study of the effects of different mutation operators and constraint handling methods, as well as other popular recombination and selection operators. Therefore, it is necessary to carry out a thorough study of the robustness and limitations of a variety of EA operators in the context of HVAC optimization problems.

Table 2.2. EA compared with common optimization methods for HVAC problems

	Deficiencies Overcome by EA
Analytical approach	EA can handle optimization problems with discrete problem variables, without existence of the derivative information of objective function and constraint functions.
Gradient-based numerical methods	Similarly to the analytical approach, the optimization problems without derivative information cannot be solved by gradient-based numerical methods. In addition, there are difficulties in determining the Hessian matrix in Newton or quasi-Newton method, and in handling the constraints by gradient-based numerical methods. EA can tackle the optimization problems in discrete variables without the presence of derivative information, it can also solve the objective function together with different types of constraints effectively.
Exhaustive search	The exhaustive search is not effective if it is used to handle multidimensional problems, since the search landscape cannot be easily previewed, but EA can conduct multidimensional search well in such circumstances. Compared to the exhaustive search, the computational cost of EA can be reduced significantly.
Direct search	EA can handle constrained problems even where the global optimum is at the boundaries of constraint functions, but this is a real limitation for direct search.
Dynamic programming	Dynamic programming can handle optimization problems which can be resolved into the required components for matrix-fill and trace-back. Dynamic programming is therefore useful for supervisory control and sequential decision problems. However for typical HVAC optimizations focused on system and plant performance, dynamic programming cannot be directly applied.

Table 2.3. EA compared with other heuristic optimization methods

	Deficiencies Overcome by EA
Simulated annealing	Apart from the existence of core but problem-specific parameters like "initial temperature" and "cooling rate", the search efficiency is also problem-dependent, since only a single individual is involved each iteration. However EA is a population-based searching process, so both the efficiency and diversity of search can be improved.
Tabu search	Tabu search is suitable to handle the spatial and network optimization problems. Apart from the existence of problem-specific parameters, tabu search involves a heavy memory load due to the tabu lists, and considerable matching search effort to avoid move reversals. This would affect the efficiency of iteration and increase the pressure of computational cost. In EA, apart from the merit of population-based exploration and exploitation, elitism would be used to maintain the best individual, only minimal memory demand is required.
Particle swarm optimization	Although particle swarm optimization (PSO) has certain similarity to the paradigms of EA due to involvement of both local and global search, it is basically different in principle. PSO relies heavily on two fundamental parameters, one related to the position of a reference particle, another to the experience of all its neighbouring particles. However problem-specific parameters are involved in the search, and they are usually not transferable. In EA, the characteristic parameters are generally transferable between different optimization problems.
Ant colony optimization	Ant colony optimization is primarily designed to tackle discrete variables rather than continuous ones. It is also difficult to incorporate constraints into the optimization process. In EA, both continuous and discrete variables can be handled, and common constraint handling techniques can be easily incorporated.

2.5 Constraint Handling Techniques for Heuristic Optimization Methods

As discussed in Section 1.2.1.3, it is common to have different kinds of constraints for the HVAC optimization problems. These constraint functions may be inequalities or equalities, and originated from the allowable ranges or physical

relationships of the problem variables. Due to the existence of constraints, the search of global optimum around the objective landscape may be more difficult, in particular the solution is near or lying on the constraints. The presence of constraints may impede the effectiveness of certain optimization methods. Different constraint handling techniques have been developed along the emerge of different optimization methods, from the classical analytical or gradient-based optimization methods to the current EA or other heuristic optimization methods.

For the classical optimization methods tackling the problem with twice continuously differentiable objective and constraint functions, the constraint handling methods like Lagrange multipliers, KT method, and sequential quadratic programming have been proven effective for the global search. However due to the development of different heuristic optimization methods, the traditional constraint handling techniques cannot be applied, since most of these optimization methods do not rely on the information of derivatives. For the heuristic and evolutionary optimization methods, Michalewicz *et al.* (1996) discussed a number of possible constraint handling approaches, which are summarized into the following approaches

- Penalizing or rejecting infeasible individuals
- Separation of objectives and constraints
- Repair of infeasible individuals and suitable replacement
- Maintaining feasible population by special representation and operator
- Use of suitable decoders to build feasible solution

The first two constraint handling approaches have been gained much attention for the heuristic and evolutionary optimization methods in the past decade, so the penalty-based method and method of separate handling objective and constraint functions are discussed

in the ensuing sections.

2.5.1 Penalty-based methods

Penalty-based methods have been applied together with a variety of heuristic optimization methods such as direct search (Lambert and Wright 1991) and GA (Asiedu *et al.* 2000, Lu *et al.* 2004) for HVAC problems. This methodology can be perceived as transforming the constrained problem into an unconstrained one, by adding a penalty to the objective function value for any infeasible individual. Then the fitness of infeasible individuals would be inevitably worse than that of feasible ones. For minimization problem with both inequality and equality constrains, provided that equality constraints are converted into inequality ones by Eq (1.3), a general penalty-based evaluation function $F_p(\mathbf{x})$ can be represented as follows:

$$F_p(\mathbf{x}) = f(\mathbf{x}) + p(\mathbf{x}) \tag{2.3}$$

where,

$f(\mathbf{x})$ = objective function

$$p(\mathbf{x}) = \sum_{j=1}^{n_{con}} \lambda_j g_j(x), \text{ penalty function} \tag{2.4}$$

$$g_j(\mathbf{x}) = \{\min[0, c_j(\mathbf{x})]\}^2, \tag{2.5}$$

constraint violation for inequality constraint $c_j(\mathbf{x}) \geq 0$

λ_j = penalty factor of $g_j(\mathbf{x})$

n_{con} = number of constraint functions, including inequality and equality constraints

This is obvious that the formulation of penalty function is the core part of different penalty-based methods, which commonly include static penalty (Homaifar *et al.* 1994), dynamic penalty (Joines and Houck 1994), annealing penalty (Michalewicz and Attia 1994) and adaptive penalty (Hadj-Alouance and Bean 1997). Their unique features are mainly associated with their corresponding penalty factors λ_j, which are summarized in

Table 2.4 below.

Table 2.4. Penalty factors of different penalty-based constraint handling methods

Method	Penalty Factor	Details
Static penalty	$\lambda_j = C_{1,j}$	$C_{1,j}$ is a constant for all $g_j(\mathbf{x})$. λ_j may be changed according to the level of violation of $g_j(\mathbf{x})$ as set by user.
Dynamic penalty	$\lambda_j = \lambda = (C_2 \cdot t)^{\alpha}$	t is number of epoch. C_2 and α are user-defined parameters (0.5 and 1 or 2 respectively as recommended by Joines and Houck 1994)
Annealing penalty	$\lambda_j = \lambda = \frac{1}{2\tau}$	$\tau = (C_3^{t-1} \cdot T_0)$, the cooling schedule, where C_3 is "cooling rate" $\in (0,1)$; T_0 is "initial temperature"; and t is number of epoch. Both C_3 and T_0 are user-defined or problem-specific parameters.
Adaptive penalty	$\lambda_j = \lambda = \lambda(t)$ where, $\lambda(t) = \begin{cases} \lambda(t-1)/\beta_1 & \text{if case(1),} \\ \beta_2 \cdot \lambda(t-1) & \text{if case(2),} \\ \lambda(t-1) & \text{otherwise.} \end{cases}$	t is number of epoch. Case(1) is situation that the best individuals in the last (t−1) epoch are always feasible, while case(2) always infeasible. The user-defined parameters β_1 and β_2 have the relationships of $\beta_1 > \beta_2 > 1$ and $\beta_1 \neq \beta_2$. Therefore in this adaptive scheme, $\lambda(t)$ would become smaller if the best individuals are feasible so far, or $\lambda(t)$ would be more serious if the preceding best individuals are all infeasible, otherwise $\lambda(t)$ remains the same as $\lambda(t-1)$.

Apart from the above four penalty-based methods that need careful design of the penalty factors, there is a very simple penalty-based method – death penalty. Death penalty is the most straightforward, any infeasible individual is simply rejected. Although the implementation is easy, it may lead to additional computational time to wait for the occurrence of feasible individuals, and it may also easily direct the search to local optimum instead of the global one.

Coello (2000b) has advocated another penalty-based method called co-evolutionary penalty, and the penalty function is defined as follows:

$$p(\mathbf{x}) = w_1 \sum_{j=1}^{n_{con}} g_j(\mathbf{x}) + w_2\, n_{viol} \qquad (2.6)$$

where,

w_1 = weighting factor for amount of violation

w_2 = weighting factor for number of violated constraints

n_{viol} = number of violated constraints

From Eq (2.6), the penalty function would consider not only the degree of constraint violation, but also the quantity of violated constraints. In the search, the whole population is divided into two sub-populations, one is used to evolve the optimum while the other is used to evolve the two weighting factors. This is why Coello (2000b) has named this to be co-evolutionary penalty.

2.5.2 Separate handling objective function and constraint violation

In the penalty-based methods, the fundamental idea is to apply penalty to the infeasible individuals so that they can be easily distinguished from the feasible ones by the penalized fitness values. Although the principle of penalty-based methods is logical, there are a variety of problem-specific or user-defined parameters which should be carefully designed to prevent from over- or under-penalizing the infeasible individuals, otherwise the search effectiveness would be adversely affected. In fact those infeasible individuals already close to the global optimum may have their role in optimal search. Therefore it would be more effective to devise suitable methods to separately consider the objective values and degree of constraint violation. The methods using this separation approach include stochastic ranking (Runarsson and Yao 2000), proposed tournament selection operator for constraint handling (Deb 2000), fitness formulation (Wright and Farmani 2001) and non-dominance method (Coello 2000a). Their features are briefly

discussed in the following sections.

2.5.2.1 Stochastic ranking

Runarsson and Yao (2000) highlighted that the over-emphasis of the penalty function in the evaluation function would deform the search landscape, and the search would be easily misled. So they devised stochastic ranking from the principle of separate handling the objective function and constraint violation. The stochastic ranking method is to arrange the rank of all individuals within population, based on their function values or degree of constraint violation by means of bubble-sort procedure. Then selection would be carried out based on these ranking results.

If the rank is fully developed in a typical bubble-sort algorithm, it is just a deterministic ranking. Therefore in stochastic ranking, a working probability p_f is assigned so that for two adjacent but competing individuals, they are allowed to compare for their function values without considering the degree of constraint violation. In order to provide this stochastic nature but prevent from over-emphasizing infeasible individuals, Runarsson and Yao (2000) suggested $0.4 < p_f < 0.5$ from their empirical study. Since it would provide less bias toward the infeasible individuals, but it still maintains the infeasible regions as potential bridges to feasible regions in the constrained search space.

2.5.2.2 Proposed tournament selection operator for constraint handling

Deb (2000) incorporated a proposed tournament selection operator for constraint handling in his real-valued (TS-R) GA optimization. In fact TS-R is not only for GA, it can be also applied in other paradigms of EA. TS-R is developed from the idea of classical tournament selection for unconstrained optimization problems. Apart from the objective function values of the opponents, TS-R would also consider their constraint violation. For any two individuals randomly picked up from the population for tournament in minimization problem, TS-R would select the winner according to one of the following three scenarios:

- If both individuals are feasible, the one with smaller objective function value will win.
- If one individual is feasible and another is infeasible, the feasible individual will win.
- If both individuals are infeasible, the one with lower constraint violation will win.

Since the two opponents are selected randomly, the infeasible individuals would still have a survival opportunity. TS-R is a constraint handling method which is free of problem-specific or user-defined parameter.

2.5.2.3 Fitness formulation

Wright and Farmani (2001) have advocated the fitness formulation method to take into account for objective function value and constraint violation separately. It is a new approach with a special design of constraint violation, called infeasibility, together with a two-part penalty function. The basic principle of fitness formulation method is to maintain the slightly infeasible individuals which have low objective function value (for minimization problem) but potential to be evolved to optimum. In the proposed

two-part penalty function, the penalty would be applied in the following situation and sequence:

- The first part of penalty is applied when there exist any infeasible individuals with objective function value lower than that of the best individual. In this case, all the infeasible individuals with objective function value higher or equal to that of the best individual would be penalized.
- The second part of penalty is to penalize all infeasible individuals in proportion to their infeasibility.

The fitness formulation method, similar to TS-R method, allows certain degree of survival of infeasible individuals and does not need any problem-specific parameters. The continual exploration of global or near-optimum would be effective.

2.5.2.4 Non-dominance method

Coello (2000a) proposed the non-dominance method which can separately handle the objective function value and constraint violation. This method firstly determines the rank for all individuals in the population through pairwise comparison. Each individual is deterministically ranked according to the following priority:

- objective function value for feasible individual;
- number of constraints being violated for infeasible individual; and
- amount of constraint violation for infeasible individual.

Then the corresponding fitness $F_i(\mathbf{x})$ of each individual is determined as follows:

$$F_i(\mathbf{x}) = \begin{cases} \text{normalized } f_i(\mathbf{x}) & \text{for feasible individual} \\ \dfrac{1}{\text{rank}_i(\mathbf{x})} & \text{for infeasible individual} \end{cases} \qquad \text{for } i = 1, \ldots, n_{pop}$$

The fitness of all individuals in population is used to construct the roulette wheel, then the individuals are selected by stochastic universal sampling. In the non-dominance method, again the user-defined or problem-specific parameters are not involved.

2.6 Summary

Different optimization methods have already been applied to solve HVAC optimization problems, including the analytical approach, gradient-based and nonlinear programming methods, exhaustive search, direct search and dynamic programming. Other emerging heuristic optimization methods may be also useful, such as simulated annealing, tabu search, particle swarm optimization and ant colony optimization. However, after careful understanding the characteristics of the three major paradigms of EA – GA; evolutionary programming and evolution strategy – they are all capable of handling the multimodal, multidimensional, nonlinear, mixed discrete-continuous and constrained problems. They are also found to be more advantageous over the traditional and other heuristic methods in tacking HVAC optimization problems. In EA, since the paradigm of GA has been involved in most of the existing research works about HVAC optimization problems, the effectiveness of evolutionary programming and evolution strategy is seldom reported. Therefore this thesis would focus on the development of EA based on these two paradigms, as well as different combinations of EA operators.

Due to the constrained nature of some HVAC optimization problems, it is also important to review the effectiveness of different constraint handling techniques, broadly categorized into the penalty-based approach and the approach of separate handling objective function and constraint violation. Different methods from these two main streams were described, and they were involved for ongoing development in this study.

CHAPTER 3 DEVELOPMENT OF PLANT SIMULATION MODELS FOR HVAC OPTIMIZATION PROBLEMS

Since the simulation-optimization approach is applied in this research work, it is necessary to develop a HVAC simulation model which can be later used for optimization purpose. In order to provide flexibility in tackling different HVAC optimization problems, a holistic component-based simulation for a centralized HVAC system is required. In this chapter, the development and details of plant simulation model of a centralized water-cooled HVAC system is presented.

3.1 Holistic Simulation of HVAC Water Side and Air Side Systems

3.1.1 Interrelationship of HVAC subsystems

A typical centralized HVAC system consists of four main subsystems. They are refrigeration plant, hydronic system, air side system and heat rejection system, as shown in Fig. 3.1. The refrigeration plant is the heart of the HVAC system, and usually there are multiple chillers in a centralized system. The hydronic system is a closed loop, delivering chilled water from the chillers to the cooling coils of air handling units (AHUs), and handles the total coil load. The air side system has a direct relationship with the conditioned space and it is relatively complex. It delivers the conditioned air from AHUs to the spaces, removes both the space sensible and latent cooling load, and returns the space air, mixes with the designed quantity of outdoor air to the AHUs. Free air cooling is applied whenever the enthalpy of outdoor air is lower than that of the space air. Lastly the heat rejection system is used to dissipate heat from the water-cooled chillers to the environment. Since the operation and energy consumption of refrigeration plant, hydronic system and heat rejection system are closely related, they are usually lumped

together and called the "water side system".

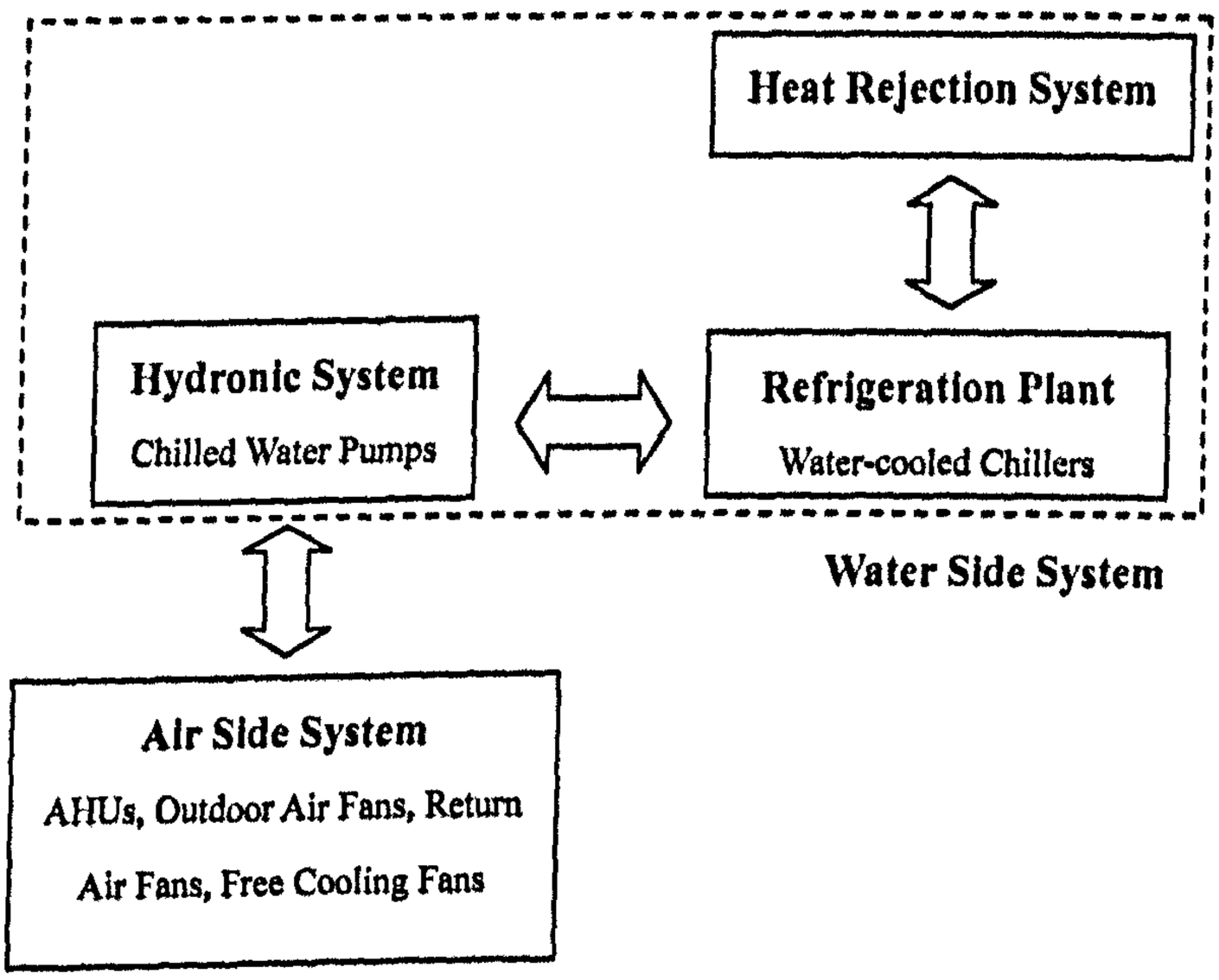


Fig. 3.1. Interrelationship among subsystems of typical centralized HVAC system

In the previous research works, the component-based plant model was seldom developed in full scale for all the subsystems of the entire HVAC system. Yik and Chan (1998), Lu *et al.* (2005a) and Lu *et al.* (2005b) developed the overall HVAC systems with component models. However, those plant models involved were represented by polynomials or biquadratic expressions in which the problem parameters were already designated, it was not flexible enough if other problem parameters were needed for optimization purposes. Therefore more detailed plant component models, particularly chiller and cooling coil, which simulate the effects of a variety of operating parameters are required for a holistic simulation of the entire HVAC system.

3.1.2 Component-based plant simulation model of entire HVAC system

A component-based model of the entire HVAC system was built up using

TRNSYS 15.3 (SEL 2000) to reflect the year-round operation of different equipment within the subsystems. The plant simulation program TRNSYS was adopted because of its high modularity and availability of a variety of HVAC plant components. Apart from the basic component models, the additional "TESS Libraries" were also included in this research. TESS stands for "Thermal Energy System Specialists", which offers updated and detailed component models for the fields of HVAC and renewable energy. These components were developed for full compatibility with the component library and simulation engine of TRNSYS. Another advantage to use TRNSYS in this research work is its input representation. TRNSYS's input file, called "deck" file (*.dck), is represented in text format, which can be conveniently linked with the optimization algorithm written by other programming language. Section 4.2.3 will discuss the coupling linkage between the plant simulation module developed by TRNSYS and the optimization module of a high-level language MATLAB.

This HVAC plant simulation model was developed with suitable reset scheme of the supply air temperature for the AHUs inside the conditioned space, as well as that of the chilled water supply temperature of the chillers. This HVAC system model was used to study the energy performance of a local subway station. Since its design is typical to the water-cooled centralized HVAC system with free cooling provision in Hong Kong, it is used to illustrate the characteristics of plant simulation model developed by TRNSYS for local application. The schematic diagram of the HVAC system and the respective plant component models is shown in Fig. 3.2.

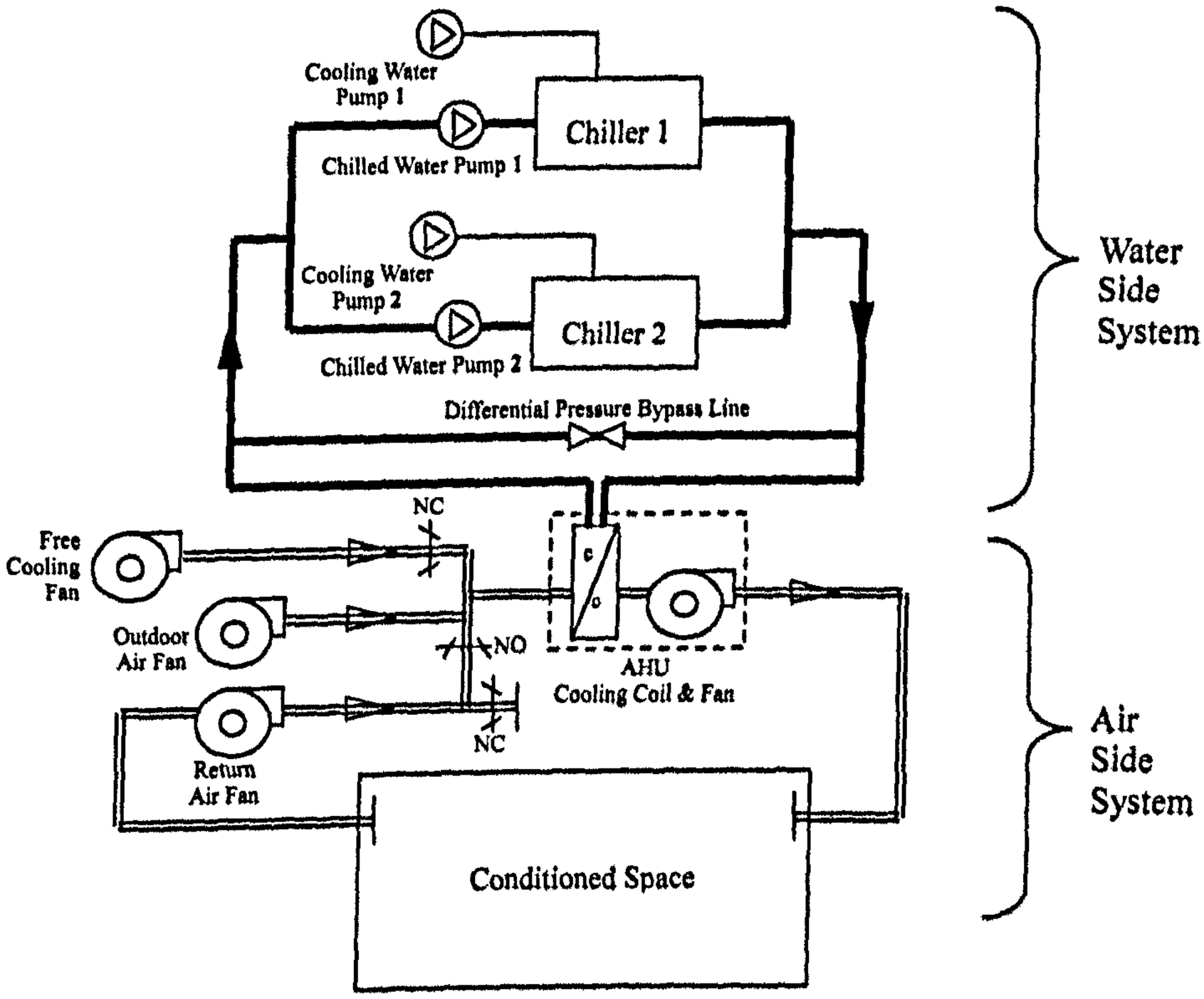


Fig. 3.2. Schematic diagram of developed HVAC simulation model

There were two chillers, with the associated chilled water and cooling water pumps. Differential pressure bypass was applied as a hydronic coupling during the part load operation. For the air side system, there were a number of AHUs and fans in the actual design. In order to simplify the model and hence save the simulation time, particularly later in the optimization process, it was assumed that the same kind of air side equipment ran in the same operating mode in parallel. Therefore it was lumped into one AHU, one outdoor air fan, one return air fan, and one free cooling fan. The screen capture of the developed HVAC simulation model in TRNSYS 15.3 is shown in Fig. 3.3. The water side system, air side system, conditioned space, inputs and output of the model are illustrated.

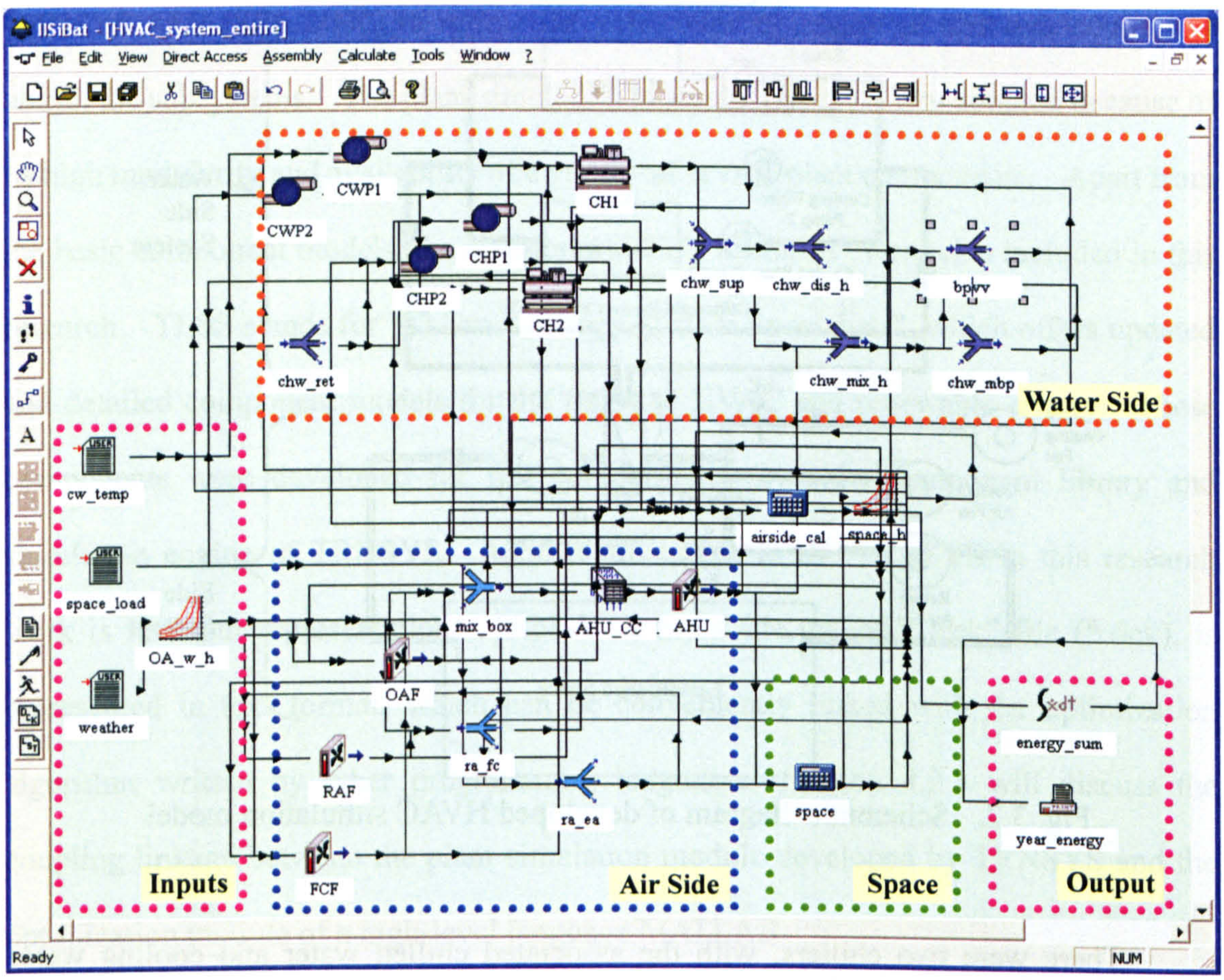


Fig. 3.3. Display of HVAC simulation model in IISiBat 3 of TRNSYS 15.3. IISiBat, "Intelligent Interface for the Simulation of Buildings", is a simulation environment program which has been adapted to house the TRNSYS simulation software. IISiBat 3 has been specially designed for working on the platform of Microsoft Windows.

3.1.3 Characteristics of component model of HVAC system

For the HVAC system, Underwood and Yik (2004) have described and summarized the major component models, which are in the forms of polynomials or biquadratic expressions. In this study, the component models available in TRNSYS were used to develop the complete HVAC system model, supported by the more detailed HVAC component models in the TESS Libraries as mentioned in Section 3.1.2. The inherent non-linearity of different component models in TRNSYS can be found from a

complex of non-linear algebraic equations involved. Among a variety of simulation components already adopted, those of the chiller and cooling coil are the most complex, and they are described in more detail below.

3.1.3.1 Water-cooled chiller model

The water-cooled chiller, TRNSYS model type 666, was based on the thermal performance and power requirement of a vapour compression refrigeration machine (TESS 2004). Two external data files were required to determine the chiller performance. The first data file consisted of two ratios as functions of the chilled water temperature set point $T_{chw,sp}$ and the cooling water entering temperature $T_{cw,i}$. They were the ratio of the actual capacity to the rated capacity G_{ratio}, and the ratio of the actual coefficient of performance to the rated one COP_{ratio}. These relationships are expressed in Eqs (3.1) and (3.2).

$$G_{ratio} = F_1(T_{chw,sp}, T_{cw,i}) \quad (3.1)$$

$$COP_{ratio} = F_2(T_{chw,sp}, T_{cw,i}) \quad (3.2)$$

The second data file contained the fraction of full load power f_{flp} as a function of the partial load ratio r_{pl} of the chiller, as expressed in Eq (3.3).

$$f_{flp} = F_3(r_{pl}) \quad (3.3)$$

The mathematical expressions used to determine the power of the water-cooled chiller are listed out in Eqs (3.4) to (3.9).

$$G = G_{rated}\, G_{ratio} \quad (3.4)$$

$$COP_{nom} = COP_{rated}\, COP_{ratio} \quad (3.5)$$

$$P_{nom} = \frac{G}{COP_{nom}} \quad (3.6)$$

$$Q_{load} = m_w C_{pw} (T_{chw,i} - T_{chw,sp}) \quad (3.7)$$

$$r_{pl} = \frac{Q_{load}}{G} \quad (3.8)$$

$$P = P_{nom} f_{flp} \quad (3.9)$$

where,

COP_{nom} coefficient of performance at nominal cooling capacity

COP_{rated} coefficient of performance at rated cooling capacity

COP_{ratio} ratio of COP_{nom} to COP_{rated}

C_{pw} specific heat capacity of chilled water at constant pressure (kJ $kg^{-1}K^{-1}$)

G_{nom} nominal cooling capacity of chiller (kW)

G_{rated} rated cooling capacity of chiller (kW)

G_{ratio} ratio of G_{nom} to G_{rated}

m_w mass flow rate of chilled water (kg s^{-1})

P_{nom} nominal chiller power (kW)

Q_{load} cooling load (kW)

T_{chw} chilled water temperature (°C)

Subscripts

i entering conditions

sp set point

After rearranging Eqs (3.1) to (3.9), the chiller power P is expressed by Eq (3.10) below. This can be found that the chiller power would have a nonlinear relationship with the chilled water supply temperature set point and cooling water entering temperature.

$$P = \frac{G_{rated} F_1(T_{chw,sp}, T_{cw,i})}{COP_{rated} F_2(T_{chw,sp}, T_{cw,i})} F_3\left[\frac{m_w C_{pw}(T_{chw,i} - T_{chw,sp})}{G_{rated} F_1(T_{chw,sp}, T_{cw,i})}\right] \quad (3.10)$$

For the outputs of this chiller component model, the actual COP, heat rejection rate, chilled water leaving temperature and cooling water leaving temperature were determined through Eqs (3.11) to (3.15) respectively.

$$Q_{met} = \text{minimum}\ (Q_{load}, G_{nom}) \tag{3.11}$$

$$COP = \frac{Q_{met}}{P} \tag{3.12}$$

$$Q_{rej} = Q_{met} + P \tag{3.13}$$

$$T_{chw,o} = T_{chw,i} - \frac{Q_{met}}{m_w C_{pw}} \tag{3.14}$$

$$T_{cw,o} = T_{cw,i} + \frac{Q_{rej}}{m_{cw} C_{pcw}} \tag{3.15}$$

where,

C_{pcw}	specific heat capacity of cooling water at constant pressure (kJ $kg^{-1}K^{-1}$)
m_{cw}	mass flow rate of cooling water (kg s^{-1})
Q_{met}	cooling load met by chiller (kW)
Q_{rej}	heat rejection rate (kW)
T_{cw}	cooling water temperature (°C)

Subscripts

i	entering conditions
o	leaving conditions

3.1.3.2 Cooling coil model

The cooling coil, TRNSYS model type 52, uses the air effectiveness model outlined by Braun (1988) and the efficiencies of rectangular-plate fins from Threlkeld (1970). The limiting exit state would be that the air was saturated at a temperature equal to that of the incoming chilled water stream. This corresponded to the minimum possible enthalpy of the exit air. The air side heat transfer effectiveness was defined as the ratio of the air enthalpy difference to the maximum possible air enthalpy difference if the exit air were at the minimum possible enthalpy. Fin efficiencies were required in order to calculate heat transfer coefficients between the air stream and the coil. On the other hand, for the local hot and humid climatic conditions in Hong Kong, usually the coil

surface temperature at the air inlet is lower than the dewpoint of the incoming air, so that the coil was completely wet during the process of dehumidification. Therefore only the wet analysis was applied for simulation, and the heat transfer rate of completely wet cooling coil was determined by Eq (3.16), with the help of Eqs (3.17) to (3.20) as follows.

$$Q_{wet} = \varepsilon_{wet}\, m_a\, (h_{a,i} - h_{s,w,i}) \tag{3.16}$$

$$\varepsilon_{wet} = \frac{1 - e^{-NTU_{wet}(1-m^*)}}{1 - m^* e^{-NTU_{wet}(1-m^*)}} \tag{3.17}$$

$$m^* = \frac{m_a\, C_s}{m_w\, C_{pw}} \tag{3.18}$$

$$NTU_{wet} = \frac{UA_{wet}}{m_a} \tag{3.19}$$

$$C_s = \frac{h_{s,w,o} - h_{s,w,i}}{T_{chw,o} - T_{chw,i}} \tag{3.20}$$

where,

C_{pw}	specific heat capacity of chilled water at constant pressure (kJ $kg^{-1}K^{-1}$)
C_s	average slope of saturation air enthalpy versus temperature (kJ $kg^{-1}K^{-1}$)
h_a	specific enthalpy of moist air (kJ kg^{-1})
h_s	specific enthalpy of saturated air (kJ kg^{-1})
m_a	mass flow rate of dry air (kg s^{-1})
m_w	mass flow rate of chilled water (kg s^{-1})
m^*	ratio of air to water effective capacitance rate for wet analysis
NTU	overall number of transfer unit
Q_{wet}	heat transfer rate of completely wet coil (kW)
T_{chw}	chilled water temperature (°C)
UA	overall heat conductance (kW K^{-1})
ε_{wet}	effectiveness of completely wet coil

Subscripts

i	entering or inside conditions
o	leaving or outside conditions
w	water stream conditions
wet	wet surface

The average saturation specific heat, C_s, depended on the outlet chilled water temperature $T_{chw,o}$ and therefore an iterative method was required to find this $T_{chw,o}$. The leaving conditions of air or chilled water passing through the cooling coil were determined through Eqs (3.21) to (3.23), with the help of Eq (3.24). Eq (3.25) was used to confirm the cooling coil was fully wet, as the coil surface temperature $T_{s,i}$ should be lower than the inlet air dewpoint.

$$h_{a,o} = h_{a,i} - \varepsilon_{wet}\,(h_{a,i} - h_{s,w,i}) \tag{3.21}$$

$$T_{a,o} = T_{s,e} + (T_{a,i} - T_{s,e})\; e^{\frac{-UA_o}{m_a C_{pa}}} \tag{3.22}$$

$$T_{chw,o} = T_{chw,i} - \frac{m_a (h_{a,i} - h_{a,o})}{m_w\, C_{pw}} \tag{3.23}$$

$$h_{s,s,e} = h_{a,i} + \frac{h_{a,o} - h_{a,i}}{1 - e^{\frac{-UA_o}{m_a C_{pa}}}} \tag{3.24}$$

$$T_{s,i} = T_{chw,o} + \frac{m_a}{m_w\, C_{pw}} \frac{UA_{wet}}{UA_i} (h_{a,i} - h_{s,w,o}) \tag{3.25}$$

where,

C_{pa}	specific heat capacity of moist air at constant pressure (kJ $kg^{-1}K^{-1}$)
T_a	air temperature (°C)
T_s	surface temperature (°C)

Subscripts

e	effective
s	surface conditions

3.2 Operation Algorithm for HVAC System

In the development of the HVAC plant simulation model, just linking up all the related components of equipment, pipework and ductwork could not simulate the real plant operation in according to different loading and climatic conditions throughout a year. Additional operation instructions were necessary, so that the operating quantity

and capacity of the equipment, as well as the corresponding flow conditions at different portions of pipework and ductwork, could be effectively determined on hourly basis. In order to provide this essential information to the plant model, a generic component for setting up control expressions was incorporated for different operation scenarios of the equipment on the TRNSYS platform. The whole plant model could then reflect the HVAC system with complete dynamic operation of different equipment and subsystems in response to the fluctuating cooling load and outdoor conditions throughout the yearly operating hours. The algorithm to determine the hourly total energy consumption of the HVAC system through dynamic operation is illustrated in Fig. 3.4.

In the hourly operation of different equipment and subsystems, there were basically three sets of input data files: total and space loads; weather conditions and cooling water temperature. For the water side system, the total cooling load was used to determine the number of chillers, in turn the numbers of the chilled water and cooling water pumps in operation. The power consumption of the chillers was determined according to the respective component model, starting from the input chilled water temperature set point and the cooling water entering temperature. For the air side system, based on the space sensible and latent loads together with the supply air temperature set point, the space air temperature and humidity ratio were found with this set point. This information was used to determine whether full speed or half speed operation was applied to the AHU fan, outdoor air fan and return air fan. In addition, enthalpy control was utilized to decide on the implementation of the free cooling mode, which would deploy the free cooling fan at the same speed of the other operating fans.

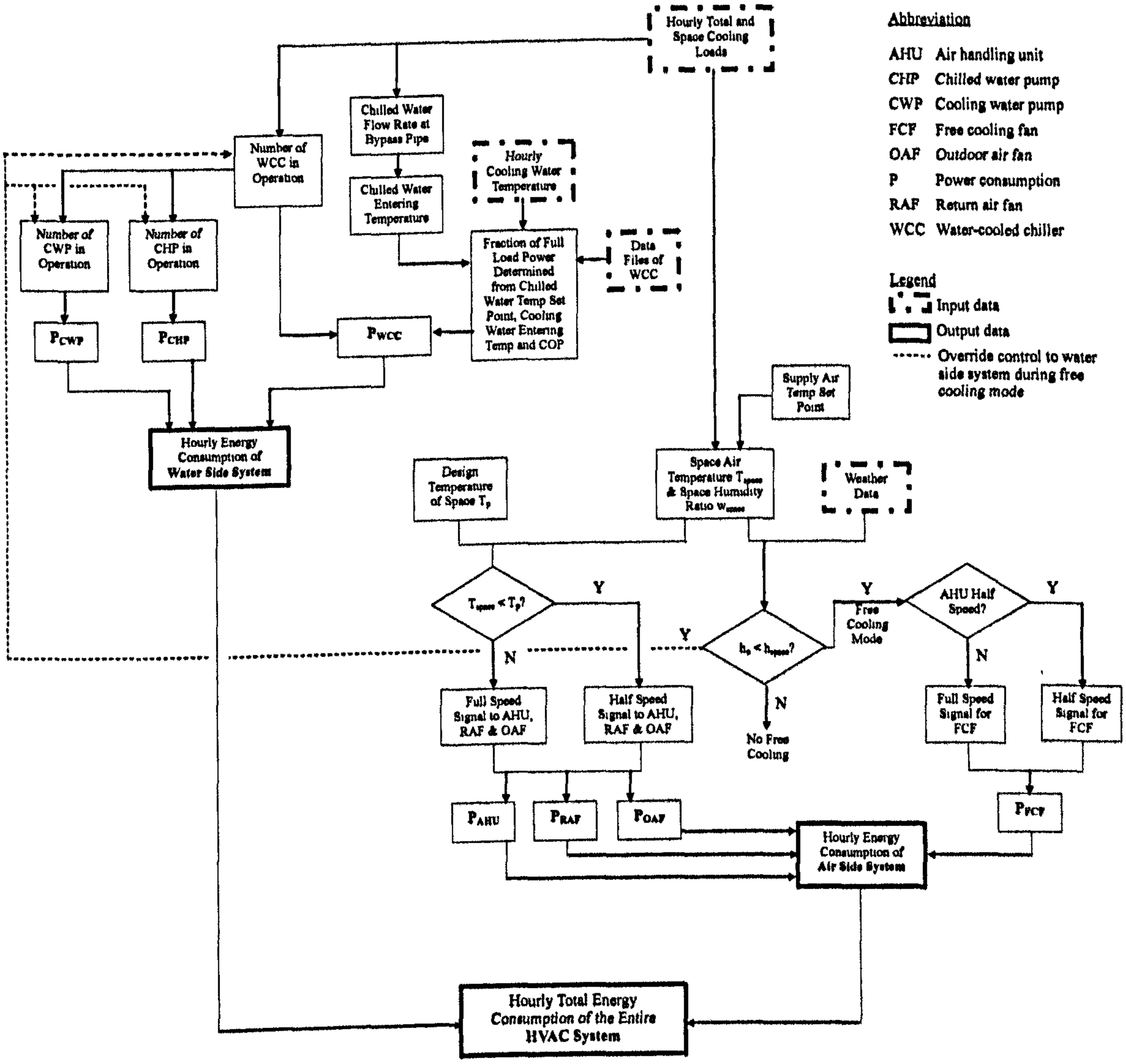


Fig. 3.4. Determination of hourly energy consumption of entire HVAC system through dynamic operation algorithm

This dynamic operation algorithm was essential to provide the necessary system response to different boundary conditions, mainly the loading and climatic conditions. And this would increase the degree of the nonlinear, intrinsically constrained and discrete nature of the system performance, in particular to the energy consumption of both the water side and air side systems. As a result, this HVAC model should be handled by an effective optimization method that could handle the nonlinear and discrete characteristics.

3.3 Possible Problem Variables and Outputs

From the developed simulation model for the entire HVAC system, there are a number of possible problem variables of the system or equipment, including:

- set point of chilled water supply temperature;
- set point of supply air temperature;
- chiller capacity;
- water flow rate of chilled water pump and cooling water pump;
- air flow rate of AHU supply air fan, return air fan, outdoor air fan and free cooling fan;
- configuration of cooling coil, such as number of rows and tubes, duct height and width, outside and inside diameters of tube, fin thickness and spacing, number of fins, tube spacing.

On the other hand, the basic optimization output for evaluation is the energy consumption. However it can cover the following aspects, and even extend to other studying areas:

- energy consumption of water side system only, or that of air side system only;
- energy consumption of single equipment, such as chiller, chilled water pump, cooling water pump, supply air fan, return air fan, outdoor air fan, free cooling fan;
- electricity cost;
- maintenance cost based on the frequency of operation;
- life cycle cost.

As a result, there is a variety of potential areas for optimization based on the developed simulation model of the holistic HVAC system. Even the number of chillers

or other equipment may be varied, it can be altered in a convenient way since the model template is available. The methodology developed in the present study can be modified or extended to other HVAC systems with increased complexity. However, it is necessary to have a balance among the depth of study, model complexity, computational resources and simulation time.

3.4 Summary

In this chapter, a holistic development of a centralized water-cooled HVAC system model is presented. The interrelationship of different HVAC subsystems, mainly between air side and water side, is explained. The entire HVAC model development was achieved by using the plant simulation program TRNSYS. The mathematical models of two complex plant components, chiller and cooling coil, are expressed in detail, and their nonlinear nature observed. To allow the deployment of the required number of HVAC equipment in response to the changing hourly cooling load, a dynamic operation algorithm was devised to determine the corresponding energy consumption of different equipment. Based on the developed HVAC simulation model, a variety of optimization problems can be studied, and it will also be used for a local application to be discussed in Section 9.2.2.

CHAPTER 4 DEVELOPMENT OF EA SUITE

After discussing the development of a component-based HVAC system model, an appropriate optimization platform is required to achieve the simulation-optimization approach in this research work. In Chapter 2, it is concluded that EA is the most suitable optimization method in handling the HVAC problems. However there are a number of EA optimization packages already available in the market, some are shareware but some are commercial products. Also discussed in Chapter 2, the paradigms of evolutionary programming and evolution strategy under EA are seldom studied as compared to GA in the context of HVAC optimization problems. The required EA optimization package should also be capable of linking up with the simulation model of HVAC system already developed by TRNSYS in Chapter 3. Therefore a thorough review of the suitability of existing EA optimization packages was carried out, and the necessity for the development of a new optimization platform is described in this chapter.

4.1 Review of Available Optimization Packages

4.1.1 GenOpt

GenOpt has been developed by Wetter (2001), and version 2.0.0 has been launched since January 2004. GenOpt is an optimization platform for minimizing an objective function that is evaluated by an external building energy or plant simulation program. From the latest user manual (Wetter 2004), there are algorithms for one-dimensional and multi-dimensional optimization, as well as a facility for parametric runs. A variety of optimization methods have been included on the GenOpt platform, including different direct search methods, discrete Armijo gradient, particle swarm optimization (PSO), and the related hybrid methods. These are mainly the

gradient-based numerical, direct search or hybrid optimization methods, except the heuristic PSO as mentioned in Section 2.2.3. For the constraint handling techniques in the current version of GenOpt, mainly penalty functions or barrier functions are used. There are not paradigms of EA nor similar heuristic approaches in this optimization package.

4.1.2 MATLAB GA and Direct Search Toolbox

MATLAB GA and Direct Search Toolbox (GADST) is an official and independent toolbox developed by MATLAB, different from the current MATLAB Optimization Toolbox 3.0.4. The latest version of GADST is 2.0.2 (R2006b), which has been released since September 2006. Apart from the implementation of GA, this toolbox also includes direct search method. This enriches its capability in handling different kinds of optimization problems. There is a "hybrid" provision which allows linkage with another optimizer for further fine-tuning the optimization results.

However, there are quite a number of limitations in GADST. For representation of the problem variables, real numbers and bit strings are included, but there is no direct representation for integer variables. For the selection operator, it follows the approach of simple GA that the selection of parents for reproduction is implemented for the variation processes. But the approach of selection for survival is not available, and the selection operator cannot be applied to offspring population in order to generate the parents for next epoch under the paradigms of evolutionary programming or evolution strategy. With respect to constraint handling, only penalty functions or Lagrange multipliers are allowed. Any customized constraint handling techniques cannot be applied or inserted in the optimization flow. If the performance of GA is evaluated with

some common test functions, multiple runs are not allowed, so extra effort is required to get statistically significant results for comparative study purposes. In addition, it is not sure whether GADST provides different initial random seeds for each run; otherwise multiple-run analyses would not be feasible.

4.1.3 Genetic and Evolutionary Algorithm Toolbox

Genetic and Evolutionary Algorithm Toolbox (GEATbx) is commercial package which has been continuously updated by Pohlheim (2005) since 1994. The latest version is 3.8, released in October 2006. GEATbx v. 3.8 can handle optimization problems with the variables in real-valued or binary representation, with the support of a variety of EA operators for recombination, mutation and selection.

If the optimization is to be carried out with the regime of GA using a commercial package, GEATbx is better than MATLAB GADST, since the former has included more choices for GA operators and greater flexibility in handling problems with different features. GEATbx has the provision that the same information can be saved into suitable files, so that the results from multiple runs can be consolidated for further analysis. However in the context of evolution programming and evolution strategy, GEATbx cannot really implement these paradigms in optimization. This is because the available options for recombination and mutation are just designed according to the paradigm of GA. Although GEATbx suggests that its features cover evolution strategy, the mutation operators do not cover the self-adapted stochastic strategy parameter commonly used in the classical evolutionary programming and evolution strategy. In addition, there are no extrinsic constraint handling techniques for tackling constrained problems All the selection operators are implemented according to the lumped fitness value already

incorporated with penalty, without the provision of separate handling objective function value and constraint violation.

4.1.4 Genetic Algorithm Toolbox for MATLAB

The Genetic Algorithm Toolbox (GAT) for MATLAB version 1.2 was developed by the Department of Automatic Control and Systems Engineering of the University of Sheffield UK in 1994, with the support of the UK SERC grant. GAT is currently a shareware (GAT 2006) and version 1.2 is still available, without any new release. This Toolbox was originally developed for MATLAB 4.2, but it can also be used in the latest MATLAB 7. Although GAT can function well on the new MATLAB platform, the operators were developed more than a decade ago, and only simple GA and multi-population GA are included. Since the development was based on the paradigm of GA, evolutionary programming and evolution strategy cannot be implemented in this optimization package.

4.1.5 Need for development of new EA Suite

After reviewing currently available optimization packages of evolutionary algorithm, the limitations and deficiencies can be summarized as follows:

a. The packages mainly focus on GA, the typical operators for evolutionary programming or evolution strategy are not included, although one of them mentions this possibility.

b. Except GenOpt, the general optimization packages do not have the provision for program call and data exchange to external programs, such as building energy or plant simulation programs. Hence these optimization packages cannot be used if the

objective functions of the optimization problems are determined by external programs; these are commonly used to build up simulation models for HVAC and engineering problems. As discussed in Section 4.1.1, although GenOpt has the capability to link up external programs including TRNSYS, its optimization methods do not include evolutionary algorithm.

c. Constraint handling techniques other than penalty-based methods are not found in those packages. There is also no provision to add other constraint handling methods, like those of separate handling objective function and constraint violation as discussed in Section 2.5.2.

As a result, it was necessary to tailor-make a new platform to overcome the aforementioned deficiencies, and this prototype of evolutionary algorithm tool, called EA Suite, was developed with the following objectives:

a. To follow the evolution framework of evolutionary programming and evolution strategy for optimization;
b. To include the necessary or newly developed EA operators;
c. To provide the coupling link and data exchange to the TRNSYS simulation program;
d. To be able to handle constrained problems with separate consideration of objective function and constraint violation;
e. To allow flexibility and modularity for continual addition of EA operators.

The details of development and implementation of this EA Suite are discussed in the following sections of this chapter.

4.2 Overview and Utilities of EA Suite

4.2.1 Programming language

In the development of this EA Suite, the programming platform MATLAB 6.1 for Windows XP was adopted. MATLAB is a high-level array language with a comprehensive control flow statements, function library, data structures and different utility features. Since the basic data element in MATLAB is an array, so dimensioning is straightforward and program development can be facilitated. This is particularly convenient in representing the population which is commonly a matrix with the dimension of $n_{pop} \cdot n_{var}$ (where n_{pop} is population size and n_{var} is number of problem variables). In addition, it includes high-level commands for two-dimensional and three-dimensional data visualization and presentation graphics. Facilities for customization of graphical appearance and development of graphical user interfaces are provided. The program interface of MATLAB can call an external executable program, which allows the EA Suite to couple with the TRNSYS program, so that the plant simulation run can be used for the evaluation process of EA optimization.

Since MATLAB is an interpreted high-level language, the advantages of convenience and user-friendliness are traded off against a relatively inefficient computational time. Since EA is a population-based probabilistic searching technique, the result from only a single run is not representative. To compare the performance of different EA operators, the number of runs should be sufficient to acquire a statistically significant result. In this situation, MATLAB cannot stack the repetitive process like the other high-level languages such as C++ or FORTRAN, and it is required to run the algorithm for each epoch anew. So it would take more execution time if the test function is complex or the number of epoch is large. However for those optimization problems

with the evaluation process actualized by an external plant simulation program (like TRNSYS in this study), the real bottleneck of computation would be solving a system of equations of the related plant model. Therefore the relative inefficiency in execution of MATLAB itself would not be critical. In Section 9.3, it is shown that the percentage of computational time due to evaluation function calls is generally greater than 95% for the practical HVAC optimization problems under study.

4.2.2 Framework of EA Suite

The skeleton of this EA Suite was built on the works of Hanby (2001, 2002a, 2002b). This EA Suite was developed to be a platform to handle the optimization-simulation problems. It was also designed to be used to test a variety of EA operators with different test functions, and to benchmark other common EA optimization methods. The key features of this EA Suite were modularity and flexibility. The major modular m-files in MATLAB included:

recomb_□.m;

mutate_□.m;

select_□.m;

const_hand_□.m; and

eval_□.m.

where "□" was the option number of the recombination operator, mutation operator, selection operator, constraint handling operator, and problem of evaluation respectively. The details of choice of problems and EA operator options are described in the following Section 4.2.2.1. Given to the modular nature, any new options could be easily extended.

4.2.2.1 Choice of problems and EA operator options

The choice of problems included the general TRNSYS simulation problem, the constrained/unconstrained test functions, and the HVAC problems for application and verification. Some of the test functions and HVAC problems had complex system of equations, which would demand significant computational cost. To involve these problems could test the robustness of the optimization method in minimizing the number of function calls. The options of different EA operators are summarized in Table 4.1. These operators basically included recombination, mutation, selection and constraint handling. The corresponding choice identifiers of different test functions and HVAC problems are summarized in Table 4.2.

Table 4.1. EA operator options

Type	Option No.	Name of EA Operator	Remark
Recombination (XO)	1	Arithmetic recombination	"99" for no recombination
	2	Uniform recombination	
	3	Geometrical recombination	
Mutation (MU)	1	Gaussian deterministic mutation	"99" for no mutation
	2	Cauchy deterministic mutation	
	3	Gaussian stochastic mutation	
	4	Cauchy stochastic mutation	
Constraint Handling (CH)	1	Infeasibility discrimination	Same effect to unconstrained problems
	2	Stochastic ranking	
	3	Dynamic penalty	
Selection (SE)	1	Ranking selection	
	2	Tournament selection	
	3	Proportional selection	

Table 4.2. Choice of optimization problems

Choice No.	Title	Remark
0	Problem developed by TRNSYS simulation model	General simulation problem in TRNSYS
1	f_{c1} (Floundas & Pardalos)	Constrained test functions with linear and/or nonlinear, equality and/or inequality constraint functions
2	f_{c2} (Himmelblau's Problem 11)	
3	f_{c3} (Floundas & Pardalos)	
4	f_{c4} (Hock & Schittkowski's Problem 113)	
5	f_{c5} (Hock & Schittkowski's Problem 100)	
6	f_{c6} (Hock & Schittkowski's Heat Exchanger Design)	
7	f_{c7} (Maa & Shanblatt)	
8	f_{c8} (Deb's Test Problem 1)	Additional constrained test functions for testing the contribution of constraint handling techniques (please refer to Sections 8.3.1.1 and 8.3.1.2 for implementation)
9	f_{c9} (Hock & Schittkowski's Problem 85)	
10	f_{c10} (Welded Beam Design Problem 1)	
11	f_{c11} (Welded Beam Design Problem 2)	
12	f_{c12} (Pressure Vessel Design Problem)	
21	f_{u1} (Sphere Model)	Unconstrained unimodal test functions
22	f_{u2} (Schwefel's Problem 2.22)	
23	f_{u3} (Schwefel's Problem 1.2)	
24	f_{u4} (Generalized Rosenbrock's Function)	
25	f_{u5} (Easom's Function)	
31	f_{m1} (Schwefel's Function 7)	Unconstrained multimodal test functions
32	f_{m2} (Generalized Rastrigin's Function)	
33	f_{m3} (Ackley's Function)	
34	f_{m4} (Generalized Griewank Function)	
35	f_{m5} (Senecal's Example Problem)	Additional unconstrained multimodal test functions for testing the micro-GA (Sections 8.2.3.2 and 8.2.3.3)
36	f_{m6} (De Jong's Stationary Multimodal Function)	
44	Design of solar water heating system	HVAC example applications
55	Energy management of subway HVAC problem	
66	Design of duct system	
77	Energy management of HVAC heat rejection system	

Remarks: The sources and details of test functions f_{c1} to f_{c12}, f_{u1} to f_{u5} and f_{m1} to f_{m6} are mentioned in Appendix.

4.2.2.2 Structure of main file

The structure of the file "main.m" is shown in Fig. 4.1. In the first epoch, the population was initialized and then evaluated. From the second to the last epoch, the population was subject to variation, evaluation and selection. The process and results were recorded in a log file, and graphical output displayed at the end of the EA run. If the TRNSYS plant simulation model was involved, the simulation file would be read before commencing the EA optimization. The functions of the major m-files are described in Section 4.2.2.3 in detail. The development and benchmarking of the specific algorithm micro-GA (MGA) is described separately in Section 8.2.2.

```
input the problem option number;
open setup file;
if problem = TRNSYS plant simulation then
        simfile.m;
fi
for run = 1:run_max do
        initialize.m;
        epoch = 1;
        eval_□.m;
        const_hand_□.m;
        select_□.m;
    print_log.m;
        for epoch = 2:epoch_max do
            if EA = 1 then
                specific algorithm of MGA (please refer to Section 8.2.2);
            else
                recomb_□.m;
                mutate_□.m;
                repair.m;
            fi
            eval_□.m;
            const_hand_□.m;
            select_□.m;
            print_log.m;
        od
od
log_plot.m;
return (f_best,avg, f_best,stdev, g_total,avg, g_total,stdev, f_best,last, x_best,last,i, g_total,last, g_last,j)
```

Abbreviation:

$epoch_{max}$	= epoch of termination
$f_{best,avg}$	= average optimal objective function value in multiple runs
$f_{best,last}$	= optimal objective function value at the last run
$f_{best,stdev}$	= standard deviation of optimal objective function value in multiple runs
$g_{last,j}$	= constraint violation of each constraint function of elite individual at last run, for $j = 1,\ldots, n_{con}$ (constrained problems only)
$g_{total,avg}$	= average total constraint violation of elite individual at $epoch_{max}$ in multiple runs (constrained problems only)
$g_{total,last}$	= total constraint violation of elite individual at the last run (constrained problems only)
$g_{total,stdev}$	= standard deviation of total constraint violation of elite at $epoch_{max}$ in multiple runs (constrained problems only)
run_{max}	= number of runs to generate statistically significant results
$x_{best,last,i}$	= elite individual of the last run for multiple runs, for $i = 1,\ldots, n_{var}$

Remark: For single run, $f_{best,last}$, $x_{best,last,i}$, $g_{total,last}$ and $g_{last,j}$ were simply the output information about the optimum. There was no output for $f_{best,avg}$, $f_{best,stdev}$, $g_{total,avg}$ and $g_{total,stdev}$.

Fig. 4.1. Structure of "main.m" of EA Suite

4.2.2.3 Functions of major m-files

There are ten major m-files arranged below according to the occurring sequence in Fig. 4.1, and the corresponding descriptions are stated as follows.

simfile.m

Reads in the "deck" file (*.dck) of the TRNSYS simulation model, particularly to identify the variables to be optimized.

initialize.m

Generates the first population according to the required number of variables n_{var} and population size n_{pop}. Each problem variable in the individual is randomly generated between its respective lower and upper bounds. Where a mutation option using a stochastic strategy parameter is adopted, the parameters are also initialized for each individual.

eval_□.m

Runs the objective function according to the problem choice number "□". For an evaluation function determined by a plant simulation model, "trnexe.exe" (the executable TRNSYS engine) is invoked by an operating system call.

const_hand_□.m

Determines the feasibility of the individuals in population through the corresponding constraint violation. In the case of ranking-based methods, the rank of individuals would be returned according to the constraint handling option number "□".

select_□.m

Selects the individuals to become the parent population for next epoch from the offspring population after recombination and/or mutation according to the selection operator "□". For constrained problems, constraint violation is also considered in the selection process. The elite individual of the offspring population is also determined.

print_log.m

Records all the individuals and their corresponding function value, total constraint violation and respective constraint values into the file "process.log". This output file would be later handled by "log_plot.m". For multiple runs, "print_log_runs.m" would be called in instead, to consolidate all the information from all runs.

recomb_□.m

Recombines the parents to generate new individual and forms a recombined population according to the recombination operator "□". The only exception is the elite individual, which is carried through to the next epoch when elitism was implemented.

mutate_□.m

Mutates each individual to become a mutated population according to the mutation option number "□", again except the elite individual would remain intact.

repair.m

Brings out-of-bound individuals back to the range between the lower and upper bounds.

log_plot.m

Consolidates the raw log data of "process.log" into a summary file, and plots the function

value against epoch on screen. For multiple runs, "log_plot_runs.m" would be called in instead, to consolidate and plot the average performance of multiple runs.

4.2.2.4 Setup file

In order to run the EA Suite according to the required choice of problem and options of different EA operators, a setup file was used to start the run of the optimization algorithm. In addition, related information of the problem, such as the population size n_{pop}, number of variables n_{var}, number of constraint functions n_{con}, bounds of problem variables, and number type of variables were included. The basic settings of the optimization run, the epoch of termination $epoch_{max}$ and number of runs run_{max} were incorporated. The typical format of a setup file is shown in Fig. 4.2, and a sample is demonstrated in Fig. 4.3 for illustration.

Title_of_Problem	
problem	(integer, choice of problem)
XO	(integer, recombination option)
MU	(integer, mutation option)
CH	(integer, constraint handling option)
SE	(integer, selection option)
n_{pop}	(integer, population size)
$epoch_{max}$	(integer, maximum epoch for termination)
n_{var}	(integer, number of variables)
n_{con}	(integer, number of constraint functions)
lb ub	(real number, lower and upper bounds of variable)
real/integer	(integer, 1 for real number; 0 for integer)
EA	(integer, 1 for micro-GA)
run_{max}	(integer, maximum run)

Fig. 4.2. Typical format of setup file

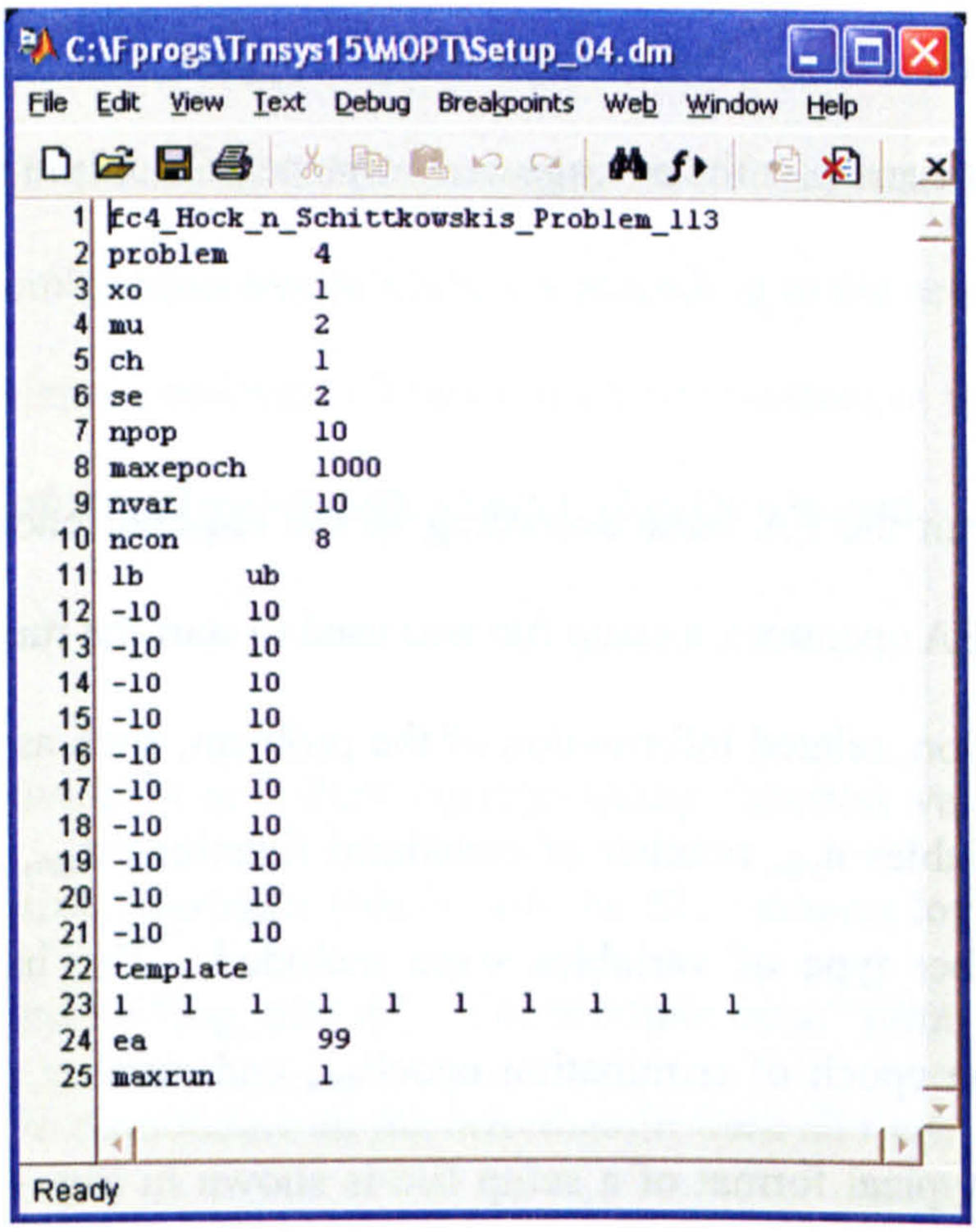


Fig. 4.3. Setup file of test function f_{c4} (Hock & Schittkowski's Problem 113)

4.2.3 Coupling capability for HVAC simulation models

This EA Suite was designed as a simulation-optimization coupling package to handle simultaneously both EA optimization and plant simulation. In this regard, the EA module and the plant simulation module were linked up together as shown in Fig. 4.4. These two modules were interconnected by a coupling linkage, in which communication was via an operating system call. The plant simulation model would provide the fitness value of the corresponding individual, then direct the results to the EA module for evaluation, selection and variation. The new population would then be sent to the plant simulation module and the loop continued until the required number of epochs reached.

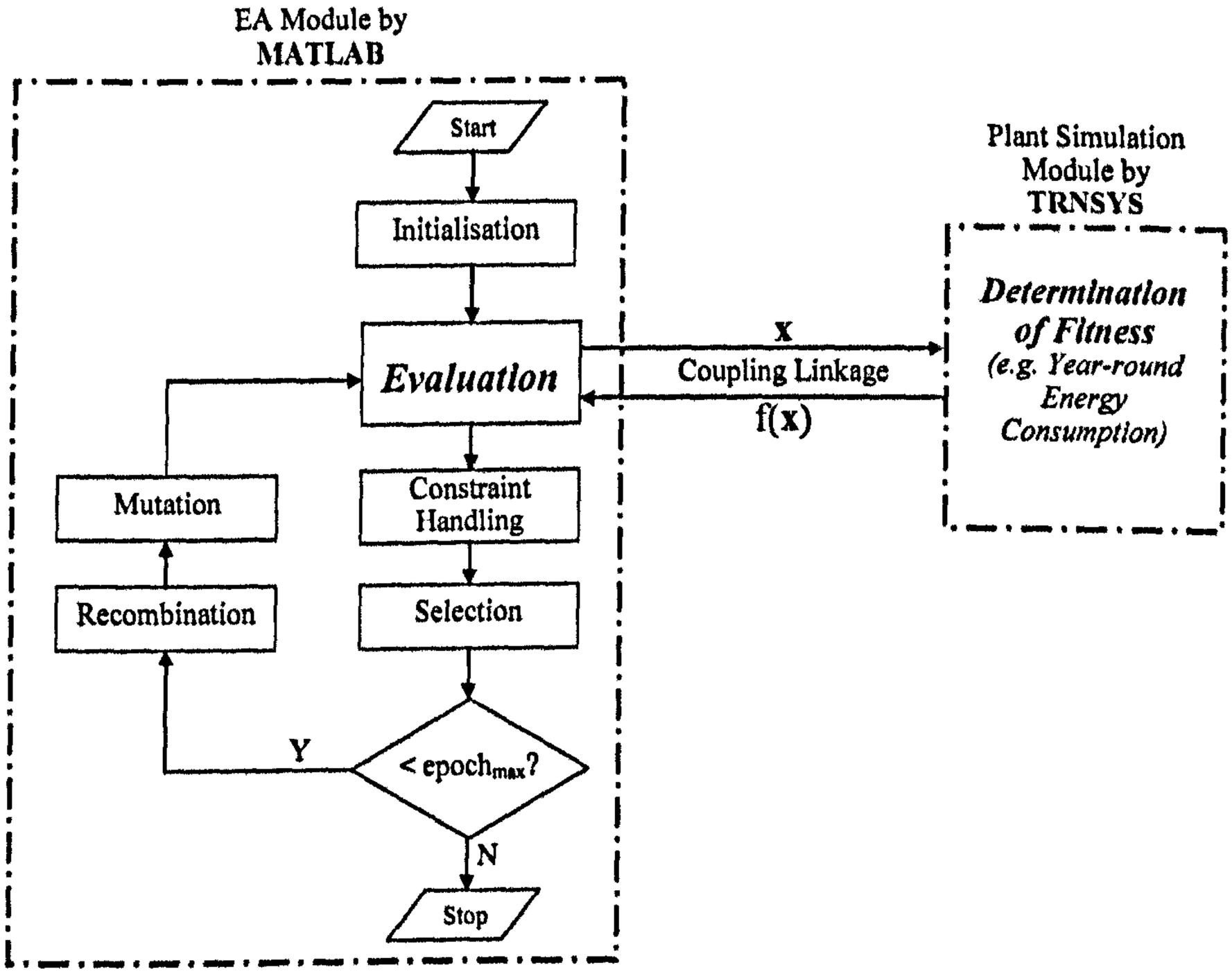


Fig. 4.4. Coupling linkage in EA Suite

For problems developed with the plant simulation model, its evaluation m-file of this EA Suite would read in the respective "deck" file, which is the standard file format containing the codes able to be executed by TRNSYS. By inserting the token "$" before the sequence number of each problem variable, like \$1, \$2, ..., $\$n_{var}$ in the "deck" file, the EA module can recognize them to be the first, second, ..., n^{th} problem variable respectively. Then the executable "trnexe.exe", the TRNSYS console engine, was called in to generate the required fitness output.

4.3 Implementation of EA Suite

4.3.1 Features of implementation

From the development of the framework to its implementation, the key features of the EA Suite can be summarized in the following aspects.

a. The EA Suite was developed according to the operating framework of evolutionary programming and evolution strategy.

b. The optimization could be implemented with or without constraint functions. For unconstrained problems, the fitness value was same as the objective function value and the process was straightforward. However for constrained problems, the EA operators of constraint handling and selection were able to consider the degree of constraint violation together with the objective function value.

c. The problem variables in real or integer number form could be handled in the EA Suite. An individual could be even a mix of both real and integer variables. One example application was the constrained energy management optimization problem of the heat rejection system described in Section 9.2.4.

d. There was provision to allow for multiple runs of the same problem, so that a set of statistically significant results would be acquired for further analysis purpose.

e. This platform was devised on the basis of single objective optimization problems. If the problem was multiobjective and its primary objective function could be singled out, this could then be handled as a single objective optimization problem with prior transformation of the other objective functions into appropriate constraint functions.

4.3.2 Start-up

Since the EA Suite is written in MATLAB, it was launched from the "main" m-file. Before starting up, the user should decide and input the required optimization parameters into the setup file. After launching the "main" m-file, a list of test problems and functions was displayed, as shown in Fig. 4.5. Then the user would be invited to input the problem number in order to call the corresponding setup file, and the

optimization process started for the respective test function or problem accordingly.

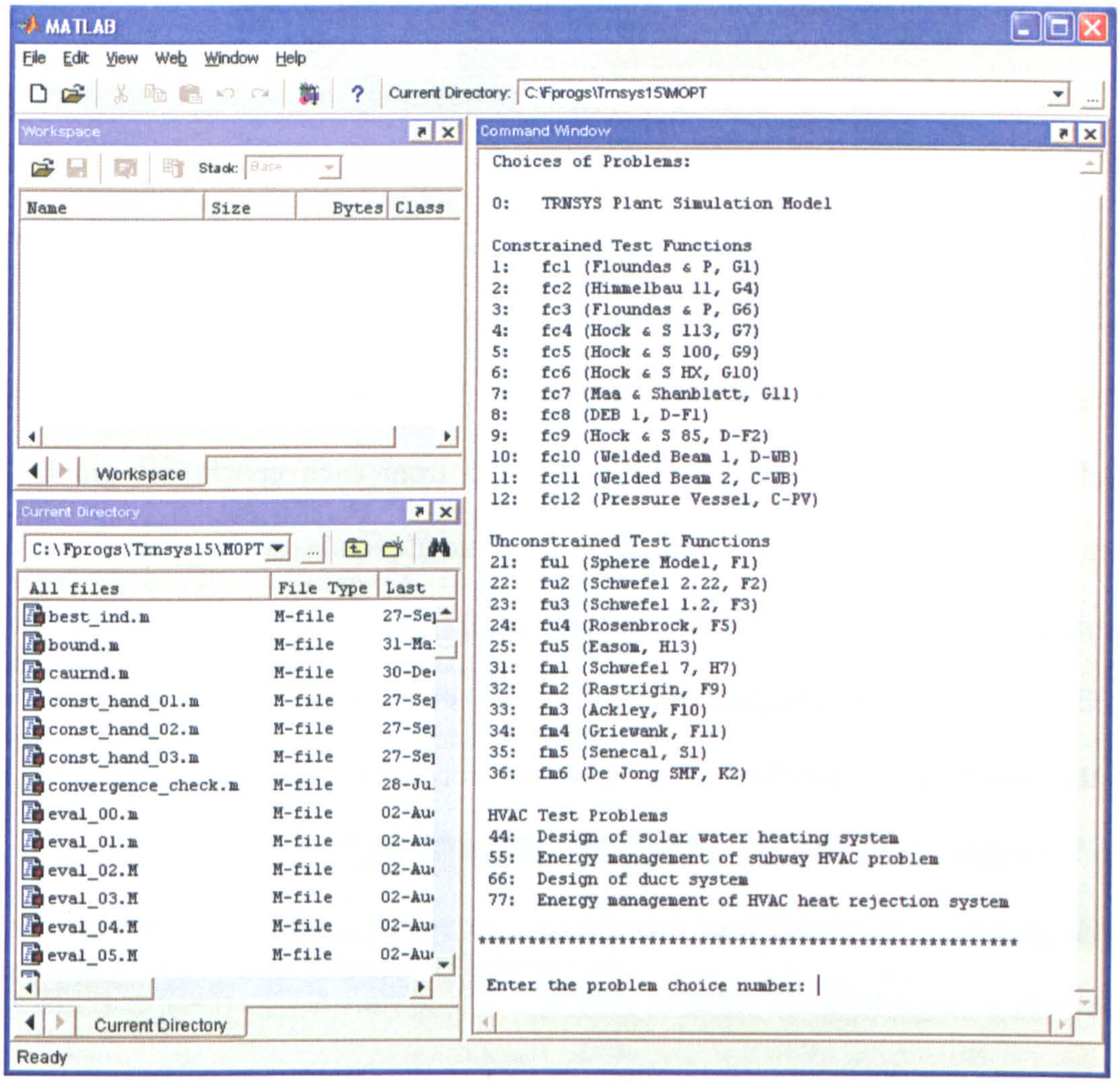


Fig. 4.5. List of test problems and functions in the Command Window of MATLAB after launching "main" m-file

4.3.3 Optimization process

At the first epoch, the population was initialized and the fitness of each individual evaluated. The whole population was carried forward to the second epoch as the parent population. Until the termination condition was satisfied, the epoch number was increased by one, the parent population would undergo the recombination process and/or mutation process. The fitness of offspring population would be evaluated, so that the new parent population can be formulated through the selection process. Generally the

termination condition would be the preset epoch of termination. This optimization process is shown in the structure of EA Suite in Fig. 4.1 and Fig. 4.4.

4.3.4 Outputs

Data recording and analysis of EA Suite was handled both for single and multiple runs. For a single run, "process.log" was the primary log file to record all the relevant data throughout the epochs, and the secondary log file "summary.log" was generated for consolidating the information of elite individual from each epoch. A graph with function value of elite f_{best} (red); mean function value of population f_{mean} (blue); and total constraint violation g_{total} (green), against epoch would be plotted and displayed accordingly. These outputs from single run are illustrated in Fig. 4.6. For multiple runs, only the information of the elite in each epoch was recorded in "process.log" for each run. Then three secondary log files, "summary_fb.log", "summary_fm.log" and "summary_gt.log" were used to contain the f_{best}, f_{mean} and g_{total} in all runs respectively. A graph with average function value of elite in multiple runs $f_{best,avg}$ (red); average mean function value of population in multiple runs $f_{mean,avg}$ (blue); and average constraint violation in multiple runs $g_{total,avg}$ (green), against epoch was plotted and displayed accordingly, as shown in Fig. 4.7. Of course for unconstrained problems, the outputs would not have any information about the total constraint violation. From the graphical results of f_{c4} as illustrated in Figs. 4.6 and 4.7, it is found that the former is relatively rugged in single run, while the latter is smoother due to averaging effect in multiple runs. In the Command Windows of MATLAB, the results at termination of a single run were listed out in Fig. 4.6, and those of multiple runs contained the statistical results of $f_{best,avg}$, $f_{best,stdev}$, $g_{total,avg}$ and $g_{total,stdev}$ in Fig. 4.7.

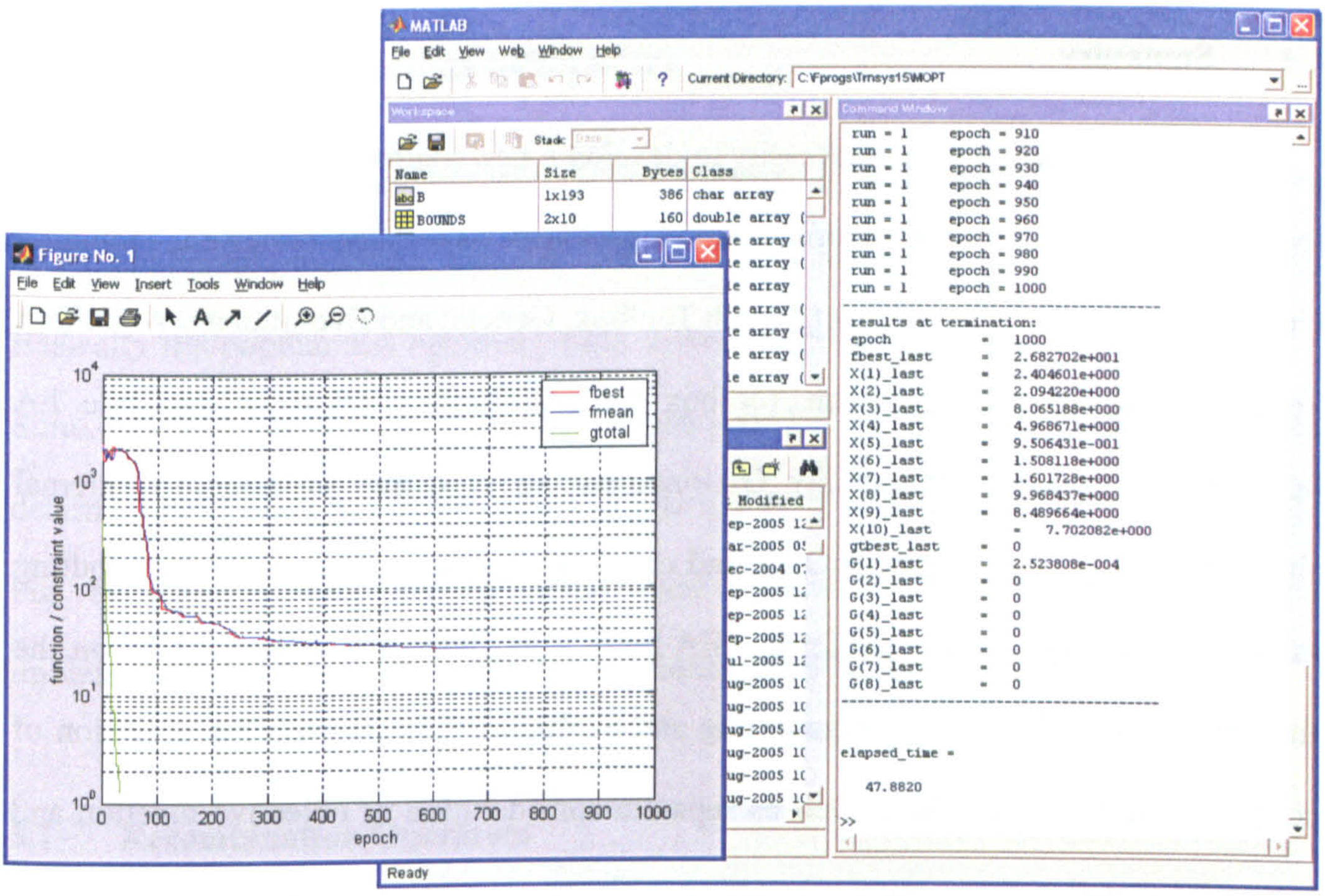


Fig. 4.6 Graphical output and results at termination of single run of test function f_{c4}

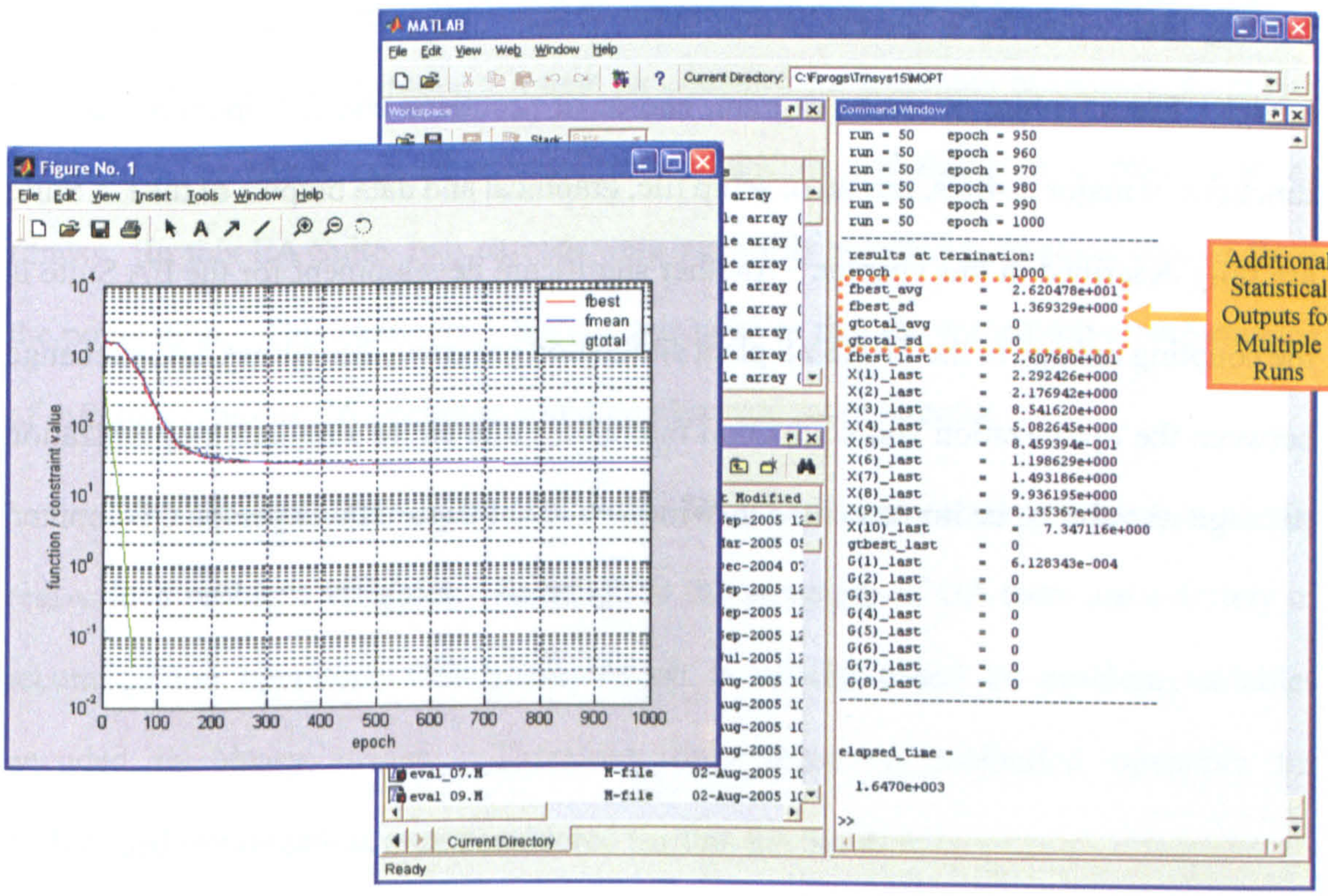


Fig. 4.7 Graphical output and results at termination of 50 runs of test function f_{c4}

4.4 Summary

In order to confirm the necessity to develop a new platform for EA optimization, the latest version of available optimization packages have been reviewed, including GenOpt, MATLAB GA and Direct Search Toolbox, Genetic and Evolutionary Algorithm Toolbox, and Genetic Algorithm Toolbox for MATLAB. All these available EA optimization packages are mainly GA-oriented, no coupling provision to external simulation program (except GenOpt), and only with penalty-based constraint handling method. Therefore, a new platform – EA Suite – has been established, based on the framework of evolutionary programming and evolution strategy, with the provision of coupling link to TRNSYS, as well as separate consideration of objective function and constraint violation for constraint handling. The programming language of EA Suite is MATLAB, its relative computational inefficiency is not a concern since most of the running time would be at the evaluation function calls of the HVAC system simulation model. The structure of main program, choice of problems and EA operator options, functions of major m-files, format of setup file, graphical and data outputs of the EA Suite are fully described in this chapter. Another significant development for the EA Suite is the coupling linkage to the TRNSYS plant simulation program, this allows data exchange between the optimization and simulation modules. The entire simulation-optimization package was able to be implemented in Windows XP of a standalone personal computer.

CHAPTER 5 EA OPERATORS OF EA SUITE

After successful establishment of the platform of EA Suite, different EA operators of recombination, mutation, selection and constraint handling can be developed there. Basically the popular EA operators from different literatures were included in the EA Suite. There are also some newly developed EA operators, including Gaussian deterministic mutation, Cauchy deterministic mutation and infeasibility discrimination. For all the involved EA operators, their principles, characteristics, roles and implementation in EA optimization are explained in detail in this chapter.

5.1 Recombination Operators

5.1.1 Introduction

Recombination is also known as crossover in the context of EA, particularly in GA. Recombination reproduces a new individual from a number of parents, from two upwards. The parent can be selected with a random probability or in relation to its fitness. In this EA Suite, two parents were randomly selected with equal chance from the population. This elementary feature was used so that the comparative performance of different types of EA operators could be observed more directly.

The recombination operators of this EA Suite were commonly used to handle real-valued problem variables. Although in the paradigm of GA there are a variety of recombination operators, their manipulation is usually based on problem variables encoded in binary strings. Therefore, only those recombination operators for real-valued optimization were considered for this EA Suite.

On the other hand, the recombined individual under this process was not the

offspring yet in this EA Suite because there were still a number of ongoing operators in the optimization process, such as mutation and selection. In the context of evolutionary programming and evolution strategy, only the population generated from the mutation operator was called offspring.

5.1.2 Arithmetic recombination

Arithmetic recombination (XO=1) was advocated by Schwefel (1981). In this recombination operator, the recombined individual $\mathbf{x}_{xo1}$ is a ratio between the two parents $\mathbf{x}_{p1}$, $\mathbf{x}_{p2}$ randomly selected from the parent population, as per Eq (5.1).

$$\mathbf{x}_{xo1} = r\,\mathbf{x}_{p1} + (1 - r)\,\mathbf{x}_{p2} \tag{5.1}$$

where $r \in \mathbf{U}(0,1)$.

In arithmetic recombination, it is obvious that $\mathbf{x}_{xo1}$ would always be generated within the search region bounded by the two parents, therefore its effectiveness in global search would be highly related to the fitness of the parents. If there is not sufficient perturbation in the course of exploration, the search outcome may be easily trapped at a local optimum.

5.1.3 Uniform recombination

Uniform recombination (XO=2) is one of the common recombination approaches in GA. However in the context of evolution strategy or evolutionary programming, the recombined individual from uniform recombination $\mathbf{x}_{xo2}$ is a mix of the variables from the two parents $\mathbf{x}_{p1}$ and $\mathbf{x}_{p2}$ (Schwefel 1981) as shown in Eq (5.2).

parents: $\mathbf{x}_{p1} = (x_{p1,1}, x_{p1,2},..., x_{p1,nvar}), \mathbf{x}_{p2} = (x_{p2,1}, x_{p2,2},..., x_{p2,nvar})$

recombined ind.: $\mathbf{x}_{xo2} = (x_{s1,1}, ..., x_{s\,nvar,nvar})$ (5.2)

where, s_i = p1 or p2 with equal chance, for i = 1,..., n_{var}

In uniform recombination, the search region is not in between the two parents, but the recombined outcome is probabilistic and scattered within the bounds of the two parents. This recombination operator can provide sufficient perturbation to avoid being trapped by a local optimum. However if both parents are feasible and close to the optimum, the uniform recombination operator is unlikely to exploit the space in between them.

5.1.4 Geometrical recombination

Geometrical recombination (XO=3) was described by Michalewicz *et al.* (1996). The recombined individual from this operator $\mathbf{x}_{xo3}$ is produced according to Eq (5.3).

$$\mathbf{x}_{xo3} = \sqrt{r\,(\mathbf{x}_{p1})^2 + (1-r)\,(\mathbf{x}_{p2})^2} \qquad (5.3)$$

where $r \in U(0,1)$, $\mathbf{x}_{p1}$ and $\mathbf{x}_{p2}$ are the two parents. Geometrical recombination is similar to arithmetic recombination in that the recombined individual would be generated within the search space between the two parents. However the major difference is that the arithmetic recombination provides a linearly intermediate individual according to the random seed, but the geometrical recombination gives a bias in the recombined individual towards the parent with higher magnitude of its variables. In addition, the nature of root-ratio-square is suitable to handle individuals with all positive real variables, or to optimize symmetrical objective functions. However in real life engineering problems, the objective functions for optimization are not likely to be symmetrical. Prior understanding of the nature of the optimization problem is necessary if this

geometrical recombination operator is to be used.

There is also a type of recombination called sphere recombination. However this is a special case of geometrical recombination with r = 0.5 only. If geometrical recombination was found to be effective, the performance of sphere recombination would also be studied.

5.2 Mutation Operators

5.2.1 Introduction

The mutation process involves a random change in the value of a problem variable, but the magnitude of the change, called step length is an important search parameter. The step length can be produced either deterministically or stochastically, with a random realization. In order to maintain the chance of unbiased perturbation, it is common to use a realization scheme with zero mean and unity variance. In general, it is common to use a Gaussian distribution in many engineering studies, however the Cauchy realization is also recommended by Yao *et al.* (1999). The Gaussian and Cauchy random numbers are generated from their respective cumulative distribution functions as discussed below.

5.2.1.1 Gaussian random number

The Gaussian (normal) probability density function is shown in Eq (5.4) as follows.

$$p_g(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right) \tag{5.4}$$

The corresponding cumulative distribution function of Gaussian probability density function cannot be determined directly, usually the cumulative function value is found by numerical methods. Michalewicz and Fogel (2004) suggested that the Gaussian random number G(0,1) (mean = 0 and variance = 1) can be generated from two independent and uniform random seeds r_1 and r_2 (where $r_1, r_2 \in U(0,1)$) in Eq (5.5) or (5.6).

$$G(0,1) = \sqrt{-2\log r_1}\sin(2\pi r_2) \tag{5.5}$$

$$G(0,1) = -\sqrt{-2\log r_1}\cos(2\pi r_2) \tag{5.6}$$

5.2.1.2 Cauchy random number

The Cauchy probability density function is shown in Eq (5.7) as follows.

$$p_c(u) = \frac{1}{\pi}\frac{t}{t^2 + u^2} \tag{5.7}$$

The corresponding cumulative distribution function of Cauchy probability density function is shown in Eq (5.8).

$$P_c(u) = \frac{1}{2} + \frac{1}{\pi}\tan^{-1} u \qquad \text{for } t = 1 \tag{5.8}$$

Therefore the Cauchy random number C(0,1) (mean = 0 and variance = 1) can be simply generated from a random seed $r \in U(0,1)$ as follows:

$$C(0,1) = \tan\left[\pi\left(r - \frac{1}{2}\right)\right] \tag{5.9}$$

Fig. 5.1 illustrates the Gaussian and Cauchy probability density functions. It can be seen that the Cauchy distribution has a longer "tail", so the Cauchy random number would potentially provide a longer step length and more prominent perturbation. This difference should result in a discrepancy in search performance.

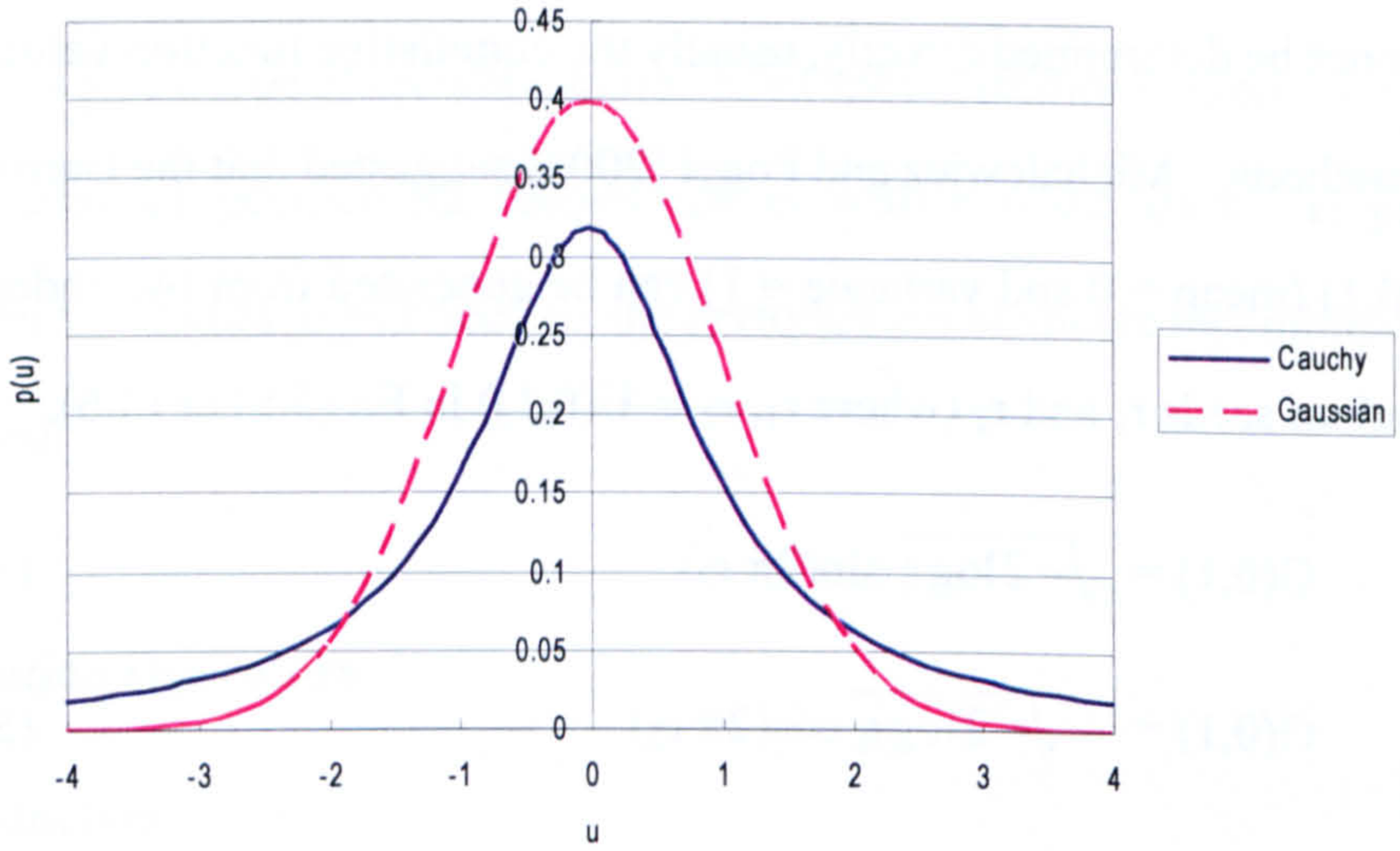


Fig. 5.1. Gaussian and Cauchy probability density functions

5.2.2 Gaussian deterministic mutation

Gaussian deterministic mutation (MU=1) has been inspired from Hanby (2001, 2002a, 2002b) and Hanby *et al.* (2005). In this mutation operator, a deterministic strategy parameter σ is the variance of the distribution and influences the step length of the mutation. The deterministic strategy parameter would be applied in Eq (5.10) for real-numbered individuals. The term of "strategy parameter" is based on the idea of self-adaptive mutation approach as adopted in the paradigm of evolutionary programming and evolution strategy (Bäck and Schwefel 1993). Since this strategy parameter is evolved in a deterministic manner as shown in Eq (5.11), so it is called "deterministic strategy parameter" in contrast to the stochastic type of strategy parameter used in classical evolutionary programming or evolution strategy.

The problem variable $x_{mul,j}$ (real number) would be mutated from the parent $x_{p,j}$ as follows:

$$x_{mul,j} = x_{p,j} + \sigma_j \, G_j(0,1) \qquad \text{for } j = 1,\ldots, n_{var} \tag{5.10}$$

where,

$$\sigma_j = \sigma_{o,j} \exp\left(-\frac{\text{epoch}-\beta}{\gamma \cdot \text{epoch}_{max}}\right), \text{ deterministic strategy parameter} \quad (5.11)$$

$$\sigma_{o,j} = \frac{ub_j - lb_j}{n_{var}^{\alpha}}$$

epoch_{max} = epoch of termination

ub_j = upper bound of the j^{th} problem variable

lb_j = lower bound of the j^{th} problem variable

n_{var} = number of variables in individual

α = 2

β = 2

γ = 0.15

$G_j(0,1)$ = Gaussian random number for the j^{th} variable of individual

The parameters α and γ were determined empirically. On the other hand, β is set at 2 in this EA Suite. From the structure of the EA Suite shown in Fig. 4.1, the first epoch is just initialization, and EA operators would take effect mainly from the second epoch. Therefore mutation would be firstly involved only from epoch 2, and it is logical to start from $\sigma_j = \sigma_{o,j}$, then β = epoch = 2 in order to produce the exponential term equal to unity.

Using a deterministic strategy parameter, this mutation operator would promote convergence of the search since the mutation step length gets exponentially smaller along the epoch as described in Eq (5.11) and illustrated in Fig. 5.2. The initial value $\sigma_{o,j}$ is set at a level directly related to the feasible range of the respective problem variable, so the actual decay rate of the deterministic strategy parameter would be different for each problem variable.

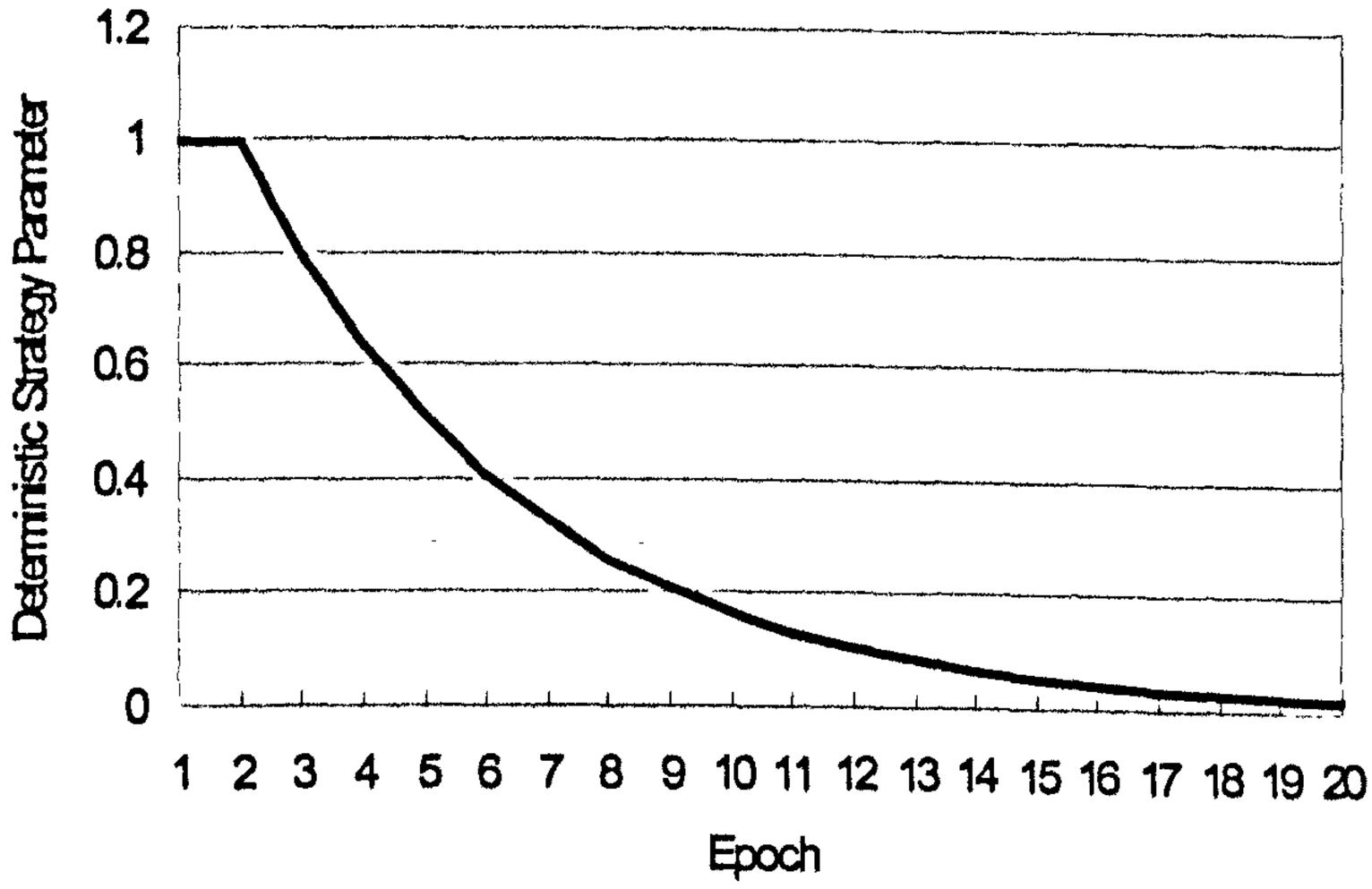


Fig. 5.2. Decay effect of deterministic strategy parameter

On the other hand for an integer problem variable $x_{mu1,j}$ a separate mutation operator is used for $x_{p,j}$ as shown in Eq (5.12).

$$x_{mu1,j} = x_{p,j} + \text{round}[G_j(0,1)] \qquad \text{for } j = 1,\ldots, n_{var} \qquad (5.12)$$

In this case, the mutation operator is used to produce a step length in integer, and it is not necessary to apply the decaying effect from the deterministic strategy parameter, otherwise the mutation effect would become stagnant when epoch goes on, since the step length would be zero after it was rounded up.

It is important to note that the global search was still stochastic in nature although a deterministic strategy parameter was adopted, since Gaussian realization G(0,1) is involved.

5.2.3 Cauchy deterministic mutation

Cauchy deterministic mutation (MU=2) was developed from the Gaussian

deterministic mutation (MU=1). The context and procedures of mutation were same as those in Section 5.2.2, except the realization effect was replaced by the Cauchy random number $C_j(0,1)$. This mutation operator carried out the variation of the individuals via Eq (5.13) or (5.14). For a real-numbered problem variable $x_{mu2,j}$, mutation from the parent $x_{p,j}$ is as follows:

$$x_{mu2,j} = x_{p,j} + \sigma\, C_j(0,1) \qquad \text{for } j = 1,\ldots, n_{var} \tag{5.13}$$

For an integer problem variable, mutation is as follows:

$$x_{mu2,j} = x_{p,j} + \text{round}[C_j(0,1)] \quad \text{for } j = 1,\ldots, n_{var} \tag{5.14}$$

where $C_j(0,1)$ is the Cauchy random number generated for the j^{th} variable of the individual **x**.

Owing to the nature of Cauchy probability density function, with a longer tail as compared to that of Gaussian as shown in Fig. 5.1, the mutation step length from Cauchy realization would have greater perturbation in the search throughout the epoch. The difference in performance is carefully studied in the experimentation in Chapter 6, in order to understand their effectiveness in EA optimization.

5.2.4 Gaussian stochastic mutation

Gaussian stochastic mutation (MU=3) was advocated by Fogel (1992) and later by Bäck and Schwefel (1993), used a stochastic strategy parameter with Gaussian realization. This mutation operator is an essential feature of classical evolutionary programming and evolution strategy. Each individual $(\mathbf{x}, \boldsymbol{\eta})$ contained both a state vector $\mathbf{x} = \{x_1, x_2, \ldots, x_{nvar}\}$ and a stochastic strategy parameter $\boldsymbol{\eta} = \{\eta_1, \eta_2, \ldots, \eta_{nvar}\}$. This pair $(\mathbf{x}, \boldsymbol{\eta})$ would be simultaneously and continuously mutated along the epochs.

For a real-numbered problem variable, $x_{mu3,j}$ and $\eta_{mu3,j}$ are mutated from the parent $x_{p,j}$ and $\eta_{p,j}$ according to Eq (5.15) and (5.16) respectively.

$$x_{mu3,j} = x_{p,j} + \eta_{p,j}\, G_j(0,1) \qquad \text{for } j = 1,\ldots, n_{var} \tag{5.15}$$

$$\eta_{mu3,j} = \eta_{p,j} \exp[\tau' G(0,1) + \tau G_j(0,1)] \tag{5.16}$$

where,

τ' $= \dfrac{K}{\sqrt{2n_{var}}}$, first learning rate

τ $= \dfrac{K}{\sqrt{2\sqrt{n_{var}}}}$, second learning rate

K $= 1$

For an integer problem variable $x_{mu3,j}$,

$$x_{mu3,j} = x_{p,j} + \text{round}[G_j(0,1)] \quad \text{for } j = 1,\ldots, n_{var} \tag{5.17}$$

In fact, mutation for a problem variable in integer shown in Eq (5.17) is identical to that of Eq (5.12) for a deterministic strategy parameter.

If the mutation operator is invoked simply as Eqs (5.15) and (5.16), premature convergence of the search process would easily occur as found in initial trial runs. In order to prevent such undesirable performance, suitable measures are needed in order to let the search continue. The following measures have been included:

a. Leveling off a minimum value for $\eta_{mu3,j}$, so that the stochastic strategy parameter would continue to provide an effective step length along the process of evolution.

b. Resetting a larger value for $\eta_{mu3,j}$ if it becomes too small. This is similar to the above measure, but the effect is more pronounced.

c. Increasing $\eta_{mu3,j}$ at the beginning of the search, so the stochastic strategy parameter

does not decay so early.

Iwamatsu (2002) suggested a limit for $\eta_{mu3,j}$ at a minimum of 10^{-4}. Ji *et al.* (2004) had a more proactive idea to reset $\eta_{mu3,j}$ back to (ub–lb)/2 (i.e. the initial value of $\eta_{mu3,j}$ in their study) if it became less than 10^{-4}. From the trial run, it was found that the idea of Ji *et al.* (2004) was more workable in preventing premature convergence. As a result, if $\eta_{mu3,j}$ is less than 10^{-4}, it is reset to the maximum allowable limit $\eta_{al,j}$ in order to reduce the chance of premature convergence. $\eta_{al,j}$ is determined as follows:

$$\eta_{al,j} = \min\left(3, \frac{ub - lb}{2}\right) \tag{5.18}$$

where the value of 3 is a typical initial stochastic strategy parameter suggested by Bäck and Schwefel (1993) for a variety of test functions. This value is used in Eq (5.18) in order to set an upper limit of $\eta_{al,j}$ if (ub – lb)/2 is larger than 3.

5.2.5 Cauchy stochastic mutation

For Cauchy stochastic mutation (MU=4), the context and procedures of mutation are same as those in Section 5.2.4, except that the Gaussian random number $G_j(0,1)$ is replaced by the Cauchy random number $C_j(0,1)$ in Eqs (5.15) to (5.17). With a real-valued problem variable, the pair $x_{mu4,j}$ and $\eta_{mu4,j}$ are mutated as follows:

$$x_{mu4,j} = x_{p,j} + \eta_{p,j}\, C_j(0,1) \qquad \text{for } j = 1,\ldots, n_{var} \tag{5.19}$$

$$\eta_{mu4,j} = \eta_{p,j} \exp[\tau' G(0,1) + \tau C_j(0,1)] \tag{5.20}$$

For an integer problem variable, mutation is as follows:

$$x_{mu4,j} = x_{p,j} + \text{round}[C_j(0,1)] \quad \text{for } j = 1,\ldots, n_{var} \tag{5.21}$$

where $C_j(0,1)$ is the Cauchy random number generated for the j^{th} variable of the

individual **x**.

To prevent premature convergence, this mutation operator is applied in the way same as the stochastic strategy parameter with Gaussian realization (MU=3).

5.3 Selection Operators

5.3.1 Introduction

In EA, selection is needed for one or more of the following scenarios:

a. Selection of individuals for next epoch, implemented for the offspring after a series of variation operations, such as recombination and mutation.

b. Selection of individuals to be the parents for recombination.

c. Selection of individuals for mutation without prior recombination (e.g. in evolutionary programming).

d. Selection of recombined individuals for mutation after recombination.

The scenario (a) is the approach of "selection for survival", while scenarios (b), (c) and (d) "selection for reproduction". In the EA Suite, the selection operator was primarily designed for scenario (a), and it was the last operator to be applied to the population in every epoch in order to produce offspring. Since elitism was applied in the EA Suite, the elite individual was identified through this operator as well. If it was the last epoch, it would generate the final population, together with the elite, for the entire EA run. On the other hand, for the scenarios (b) to (d), selection was used before executing the related variation operator, following the paradigm of genetic algorithm. Since the EA Suite was developed using the paradigms of evolution strategy and evolutionary programming, so the selection operator would be sequenced according to scenario (a).

On the other hand, the scenario (d) would not be appropriate for the EA Suite, since the function value of newly recombined individuals should be found through the evaluation process before the selection for mutation was implemented; however this would inevitably double the frequency of evaluation in each epoch. In the HVAC optimization problems under current study, function evaluation was realized by running the TRNSYS component-based plant simulation models. Evaluation was the most time-demanding stage, and the robust design of EA should therefore provide satisfactory results with minimum evaluation function calls as far as possible.

5.3.2 Proportional selection

Proportional selection (SE=3) is a common stochastic approach to select the new population from offspring in EA. In the EA Suite, the pseudo-code of proportional selection is shown in Fig. 5.3.

```
if epoch > 1 then
        develop probability vector;
        preserve the best individual from parent population;
        conduct roulette wheel to select for new population;
fi
return the elite individual from new population;
determine mean fitness value of new population;
```

Fig. 5.3. Structure of proportional selection or ranking selection

The probability vector contains the relative fitness of each individual, which is developed from the objective function values according to Eq (5.22).

$$\left(\frac{\frac{1}{f_1}}{\sum_{i=1}^{n_{pop}}(1/f_i)} \quad \frac{\frac{1}{f_2}}{\sum_{i=1}^{n_{pop}}(1/f_i)} \quad \cdots \quad \frac{\frac{1}{f_{n_{pop}}}}{\sum_{i=1}^{n_{pop}}(1/f_i)} \right) \tag{5.22}$$

This mimics the roulette wheel in that it would give higher probability for the individual with better (i.e. lower for minimization) function value, hence there would be a correspondingly higher chance for the better individuals to be selected.

To return the elite individual from the new population, the subroutine in Fig. 5.4 is adopted.

```
if [unconstrained problem] then
        return the elite individual by function value;
else (i.e. constrained problem)
        if [existence of feasible individuals] then
                return the elite individual from feasible ones by function value;
        else (i.e. all infeasible individuals)
                if [infeasibility discrimination] then
                        return elite individual by infeasibility;
                else
                        return elite individual by handling both function value &
                                total constraint violation;
                fi
        fi
fi
```

Fig. 5.4. Procedure of returning elite individual

5.3.3 Ranking selection

Ranking selection (SE=1) is a stochastic approach to select the new population from offspring in EA regime. It is similar to proportional selection as mentioned in

Section 5.3.2, but its probability vector is developed with reference to the rank of individuals instead of the fitness of individuals used in proportional selection. The pseudo-code of this selection operator is same as Fig. 5.3. The relative fitness of each individual is developed from its rank in population, instead of its objective function value as follows:

$$\frac{n_{pop}+1-rank_i}{\sum_{i=1}^{n_{pop}} rank_i} \tag{5.23}$$

where $rank_i$ is rank of the i^{th} individual, for $i = 1,\ldots, n_{pop}$.

The probability vector for roulette wheel selection contains the fitness developed from Eq (5.23). Therefore the design of this roulette wheel would result higher probability for the individual with higher rank to be selected from the offspring. After the new population was produced, the elite individual was identified from a subroutine with the pseudo-code in Fig. 5.4.

5.3.4 Tournament selection

Tournament selection (SE=2) is another common approach to select the new population from offspring in EA. The pseudo-code of tournament selection is shown in Fig. 5.5.

```
if epoch > 1 then
        unite the parents and offspring population;
        conduct pair-wise comparison from preset number of opponents;
        select the individual with the most wins;
fi
return the elite individual from new population;
determine mean fitness value of new population;
```

Fig. 5.5. Structure of tournament selection

In tournament selection, pair-wise comparison is conducted for all individuals (i.e. 2 n_{pop}) in the union of parent and offspring population. For each individual, a preset number of q opponents is randomly chosen within the union for comparison purpose, where,

$$q = \min\left\{\left[\left[\text{round}\left(\frac{n_{pop}}{2}\right)\right]\right], 10\right\} \tag{5.24}$$

In each comparison, if the fitness of the individual is better or equal to that of the opponent, it would receive a "win". Finally n_{pop} individuals out of the union are selected by counting the most "wins", hence becoming the new population.

To return the elite individual and mean fitness value from the new population, the subroutines as shown in Fig. 5.4 is adopted.

5.4 Constraint Handling Operators

5.4.1 Introduction

The general pseudo-code of the constraint handling operator in the EA Suite is shown in Fig. 5.6. In the following Sections 5.4.2 to 5.4.4, the corresponding features of

the constraint handling technique are discussed. In the EA Suite, apart from stochastic ranking, a rank-based approach was also introduced for the other two kinds of constraint handling operators. However, the rank was determined by a deterministic approach, instead of the probabilistic approach used in stochastic ranking.

The constraint handling operators in the EA Suite were developed using a rank-based rather than a value-based approach. In a population, the individual under the value-based constraint handling approach would be selected directly according to its fitness value. But under the rank-based approach, the rank could be arranged from the fitness value or some other method, such as the bubble-sort in stochastic ranking or degree of infeasibility in the operator of infeasibility discrimination. Since the constraint handling operators in the value-based approach would have a dominating effect if the values for certain individuals were much larger than the others, this would give a biased selection for those individuals. As a result, the rank-based scheme was adopted for constraint handling in the EA Suite.

For all the constraint handling operators, a rank as assigned to individuals for both unconstrained and constrained problems, as shown in Fig. 5.6. Basically this was used for constrained problems. For unconstrained problems, the rank of individuals was also generated, but simply according to the function value and independent of the option number of constraint handling.

if [unconstrained problem] **then**

return rank of individuals by their function values;

else (i.e. constrained problem)

preserve the elite individual;

determine g_{ij} and $g_{total,i}$ for the rest of population;

count and return number of infeasible individuals from offspring based on $g_{total,i}$;

return rank of offspring according to specific constraint handling technique;

fi

(where g_{ij} and $g_{total,i}$ are violated constraint value and total constraint value respectively, for $i = 1,\ldots, n_{pop}$; $j = 1,\ldots, n_{var}$.)

Fig. 5.6. General structure of constraint handling operator in EA Suite.

The returned number of infeasible individuals was the offspring population after the process of variation, not the new population after selection.

5.4.2 Dynamic penalty

The dynamic penalty constraint handling operator (CH=3) was developed from the work of Joines and Houck (1994) as mentioned in Section 2.5.1. The fitness value of the i^{th} individual $f_{dp,i}$ is determined by adding a dynamic penalty to the function value f_i. In this operator of the EA Suite, no problem-specific penalty factor is introduced in order to prevent the problem-dependent need. The dynamic penalty would have an increasing effect along the epoch as follows:

$$f_{dp,i} = f_i + \underbrace{\text{epoch} \cdot g_{total,i}}_{\text{dynamic penalty}} \qquad \text{for } i = 1,\ldots, n_{pop} \tag{5.25}$$

where $g_{total,i}$: total constraint violation of the i^{th} individual, determined by:

$$g_{total,i} = \sum_{j=1}^{n_{con}} g_{ij} \tag{5.26}$$

and,

g_{ij} : violated constraint value of j^{th} constraint function due to the i^{th} individual, determined as follows:

$$g_{ij} = \text{abs}(c_{ij}) \quad \text{for } c_{ij} < 0, \quad j = 1,\ldots, n_{con} \tag{5.27}$$

$$\text{or} \quad g_{ij} = 0 \quad \text{for } c_{ij} \geq 0, \quad j = 1,\ldots, n_{con} \tag{5.28}$$

c_{ij}: raw constraint value directly determined from the j^{th} constraint function for the i^{th} individual

5.4.3 Stochastic ranking

Stochastic ranking (CH=2) was developed with reference to the work of Runarsson and Yao (2000) as mentioned in Section 2.5.2.1. In this scheme $g_{total,i}$ was determined according to Eqs (5.26) to (5.28). The rank of any adjacent individuals was determined through a stochastic bubble-sort scheme as shown in Fig. 5.7.

for $j = 1$ to n_{pop} **do**
  for $k = 1$ to $(n_{pop} - 1)$ **do**
    if $[g_{total(rank(k))} = 0$ & $g_{total(rank(k+1))} = 0]$ or $[r < p_f]$ **then**
      if $f_{(rank(j))} > f_{(rank(j+1))}$ **then**
        swap rank of k^{th} and $(k+1)^{th}$ individuals;
      fi
    else
      if $g_{total(rank(k))} > g_{total(rank(k+1))}$ **then**
        swap rank of k^{th} and $(k+1)^{th}$ individuals;
      fi
    fi
  od
od

(where r is random number; p_f is probability to proceed stochastic ranking for not both feasible individuals, and $p_f = 0.45$ according to Runarsson and Yao 2000.)

Fig. 5.7. Procedure of bubble-sort of stochastic ranking

5.4.4 Infeasibility discrimination

The constraint handling operator infeasibility discrimination (CH=1) was developed from both the works of Deb (2000) and Wright and Farmani (2001) as described in Sections 2.5.2.2 and 2.5.2.3 respectively. The approach to determine the rank is different for the two scenarios: existence of feasible individual; or all infeasible individuals in the population. The implementation of this operator in different scenarios is discussed in the ensuing sections.

5.4.4.1 Determination of rank for existence of feasible individuals in population

If there are any feasible individuals in the population, they might be all feasible individuals or a mix of feasible and infeasible individuals. In order to distinguish fitness between the feasible and infeasible individuals, Deb (2000) devised the following approach for this purpose:

$$f_{sel,i} = f_i \qquad i = 1,\ldots, n_{pop}, \text{ for feasible individual} \quad (5.29)$$

$$\text{and} \quad f_{sel,i} = f_{max} + g_{total,i} \qquad i = 1,\ldots, n_{pop}, \text{ for infeasible ind.} \quad (5.30)$$

where,

$f_{sel,i}$: fitness value of the i^{th} individual (smaller the better for minimizaiton)

f_i: objective function value of the i^{th} individual

f_{max}: maximum objective function value of population

$g_{total,i}$: total constraint violation, determined by Eqs (5.26) to (5.28)

Since the fitness value of all the individuals are assigned with consideration of both their feasibility and constraint violation, their rank within the population was then arranged deterministically.

5.4.4.2 Determination of rank for entire infeasible population

This constraint handling approach was developed from the work of Wright and Farmani (2001), and the original idea to formulate the infeasibility of individuals for a population was adopted. If all the individuals are infeasible in the population, identification of the degree of infeasibility is required. The infeasibility of the i^{th} individual $infy_i$ is determined for each individual based on the corresponding constraint value(s) as follows:

$$infy_i = \frac{\sum_{j=1}^{n_{con}} \frac{g_{ij}}{g_{ij,max}}}{n_{con}} \qquad i = 1,\ldots, n_{pop} \qquad (5.31)$$

where $g_{ij,\,max}$ is the maximum violated constraint value among the j^{th} violated constraint value g_{ij} across the population.

Larger values of $infy_i$ show a higher degree of infeasibility, so the rank would be based on $infy_i$ accordingly.

5.5 Supplementary Strategies and Parameters

In the EA Suite, the core operators of EA optimization include recombination, mutation and selection. For constrained problems, a suitable constraint handling operator was also involved. However in order to guarantee the continuation and success of global search, certain supplementary EA operational strategies and parameters should be incorporated. These supplementary operators included elitism, repair algorithm and probabilities of recombination and mutation.

5.5.1 Elitism

Elitism means to retain the best individual for the offspring population. Sometimes not just the best one would be kept, but the second best or third best would be chosen as well. In an EA with a high population size (e.g. in the order of hundreds in GA), elitism is not commonly used. Since a large population would already include a number of potentially optimal individuals, they have a higher chance of being carried forward to the next. Nevertheless in a small population size, there may be an opportunity that the elite would not be selected under a stochastic selection operator. In this case, elitism is important to enhance the search effectiveness.

In the development of the EA Suite, elitism was therefore introduced to maintain the success of the continual global search. Since the number of evaluation function calls was critical in the current study, the population size would not be large in order to minimize the frequency of function evaluation. Therefore a single elite, i.e. only one best individual, was identified and maintained from the selection process. The elite would not just have the lowest function value for the minimization problem, but it should have the minimum or even no total constraint violation for the constrained problem. Therefore the elite individual was chosen according to Fig. 5.4 in Section 5.3.2.

Due to the existence of an elite in the stages of recombination and mutation, the number of recombined or mutated individuals would be the population size less one. Therefore the total number of evaluation function calls n_{eval} would be no longer simply $n_{pop} \cdot epoch_{max}$, but it could be reduced and determined from Eq (5.32) below.

$$\begin{aligned} n_{eval} &= n_{pop} + (n_{pop} - 1)\,(epoch_{max} - 1) \\ &= epoch_{max}\,(n_{pop} - 1) + 1 \end{aligned} \tag{5.32}$$

This shows that from the second epoch, the number of evaluation function call would be reduced once, since the fitness value of the elite was already in memory. This would save simulation time and enhance the overall optimization efficiency.

5.5.2 Repair scheme

The bounds on the problem variables for each individual were established in the setup file. During the evolution, particularly after the mutation process, some individuals would have their elements out of bounds due to the finite step size. The provision of repair was applied for such individuals so that they could be moved back to the required bounds. Based on this idea, there would be a number of approaches for repairing:

a. Restoring any out-of-bound problem variable back to its bound, and the other "good" variables remained intact.

b. Reinitializing the out-of-bound element within the bounds, and the others remained intact.

c. Rejecting the whole individual, and reinitializing a brand new individual.

Since it would be possible to have the global optimum along one or more bounds, if the search was approaching to the optimum, there would inevitably be a certain number of individuals out of bounds after variation, particularly so in the case of mutation, therefore the first approach was adopted in the EA Suite. The whole population would be inspected in the repair subroutine after mutation (or recombination if mutation was not involved). Any problem variable of the out-of-bound individual $x_{out,j}$ (for $j = 1, \ldots, n_{var}$) was restored to the closest bound according to the pseudo-code of the repair algorithm in Fig. 5.8 below.

```
if x_out < x_lower bound then
        x_out = x_lower bound;
elseif x_out > x_upper bound then
        x_out = x_upper bound;
fi
```

Fig. 5.8. Repair subroutine in EA Suite

5.5.3 Probability of recombination and mutation

In the paradigm of genetic algorithm, the probabilities or fractions of recombination (crossover) and mutation are also problem-dependent parameters. It is a common practice that the recombination probability is above 0.5, and the mutation probability 1/L, where L is the chromosome length. However L is also problem-specific and related to the resolution of solution. But in the paradigms of evolutionary programming and evolution strategy, all the individuals would undergo the process of recombination and/or mutation, so the probabilities of both recombination and mutation become unity.

5.6 Summary

There are four major types of operators in the EA Suite – recombination, mutation, selection and constraint handling. Supplementary strategies to assist the global search are also included. For the key EA operators, recombination facilitates exploitation in global search, mutation is responsible for exploring new search opportunity, selection determines the new population for next epoch, and constraint handling assists the selection in the constrained problems by ranking the feasible offspring, as well as the potential but infeasible one. The EA operators of the EA Suite are summarized below:

a. Recombination operators

- Arithmetic recombination (XO=1)
- Uniform recombination (XO=2)
- Geometrical recombination (XO=3)

b. Mutation operators

- Gaussian deterministic mutation (MU=1)
- Cauchy deterministic mutation (MU=2)
- Gaussian stochastic mutation (MU=3)
- Cauchy stochastic mutation (MU=4)

c. Selection operators

- Ranking selection (SE=1)
- Tournament selection (SE=2)
- Proportional selection (SE=3)

d. Constraint handling operators

- Infeasibility discrimination (CH=1)
- Stochastic ranking (CH=2)
- Dynamic penalty (CH=3)

e. Supplementary strategies

- Elitism
- Repair scheme

CHAPTER 6 EXPERIMENTAL SETUP AND PRELIMINARY RUNS

Regarding the highly constrained optimization problems of HVAC systems, there are generally a number of constraint functions which can be of an equality or inequality, linear or non-linear nature. On the other hand for the HVAC optimization problems developed by plant simulation models, the involved interaction constraint functions are commonly already included in the respective objective function that is represented by the simulation model. So these types of optimization problems may not have any extrinsic constraint functions, and they become unconstrained. Owing to this, both constrained and unconstrained test functions should be incorporated in the test runs in order to derive suitable configurations of EA for different HVAC optimization problems. In this chapter, the development of experimentation is presented. Since an exhaustive analysis of all combinations in this empirical study can be prevented, a series of preliminary runs was therefore implemented to screen out the less effective EA operators before launching the full runs.

6.1 Test Functions

It is a standard methodology for the researchers in evolutionary algorithms to use commonly established test functions to conduct empirical studies into the effectiveness of any newly developed EA operators (Yao *et al.* 1999, Deb 2000, Wright and Farmani 2001, Iwamatsu 2002, Ji *et al.* 2004). In order to have an in-depth study of the appropriate operators and combinations in EA for the HVAC optimization problems, a variety of constrained and unconstrained test functions were used in this research and they are listed, together with a summary of their characteristics in Table 6.1. For constrained test functions, linear inequality, non-linear equality and non-linear inequality were involved.

In the case of unconstrained test functions, five have a unimodal nature while four are multimodal. The characteristics of the objective functions include linear, quadratic, cubic, polynomial, trigonometric, and exponential types. The choice of these test functions was targeted on covering the range of formats of the optimization problems that are encountered in HVAC design and energy management.

Table 6.1. Summary of characteristics and origins of test functions

Category	Test function	n_{var}	Type of objective function	No. of constraints	Origin
Constrained problems	f_{c1} Floundas & P	13	Quadratic	LI: 9	Floundas and Pardalos (1987)
	f_{c2} Himmelblau 11	5	Quadratic	NI: 6	Problem 11, Himmelblau (1972)
	f_{c3} Floundas & P	2	Cubic	NI: 2	Floundas and Pardalos (1987)
	f_{c4} Hock & S 113	10	Quadratic	LI: 3 NI: 5	Problem 113, Hock and Schittkowski (1981)
	f_{c5} Hock & S 100	7	Polynomial	NI: 4	Problem 100, Hock & Schittkowski (1981)
	f_{c6} Hock & S HX	8	Linear	LI: 3 NI: 3	Heat Exchanger Design Problem, Hock and Schittkowski (1981)
	f_{c7} Maa & Shanblatt	2	Quadratic	NE: 1	Maa and Shanblatt (1992)
Unconstrained unimodal problems	f_{u1} Sphere Model	10	Quadratic	---	Sphere Model
	f_{u2} Schwefel 2.22	10	Polynomial	---	Schwefel's Problem 2.22
	f_{u3} Schwefel 1.2	10	Quadratic	---	Schwefel's Problem 1.2
	f_{u4} Rosenbrock	10	Polynomial	---	Generalized Rosenbrock's Function
	f_{u5} Easom	2	Trigonometric & exponential	---	Easom's Function
Unconstrained multimodal problems	f_{m1} Schwefel 7	10	Trigonometric	---	Generalized Schwefel's Problem 2.26
	f_{m2} Rastrigin	10	Quadratic & trigonometric	---	Generalized Rastrign's Function
	f_{m3} Ackley	10	Exponential & trigonometric	---	Ackley's Function
	f_{m4} Griewank	10	Quadratic & trigonometric	---	Generalized Griewank Function

Remark: LI: Linear inequality; NE: Non-linear equality; NI: Non-linear inequality

6.2 Design of Experiments

The possible series of empirical studies was designed according to the working combinations in Figs. 6.1 and 6.2, as well as the parameters of experimentation in Table 6.2.

PROBLEM		XO		MU		CH		SE
f_{c1}								
f_{c2}		1		1				
f_{c3}		2		2		1		1
f_{c4}	×	3	×	3	×	2	×	2
f_{c5}		99		4		3		3
f_{c6}				99				
f_{c7}								

Fig. 6.1. Empirical study for constrained test functions

PROBLEM		XO		MU		CH		SE
f_{u1}								
f_{u2}								
f_{u3}		1		1				
f_{u4}		2		2		any		1
f_{u5}	×	3	×	3	×	option	×	2
f_{m1}		99		4				3
f_{m2}				99				
f_{m3}								
f_{m4}								

Fig. 6.2. Empirical study for unconstrained test functions

Abbreviation

XO=1 Arithmetic recombination
XO=2 Uniform recombination
XO=3 Geometrical recombination
XO=99 (no recombination)
MU=1 Gaussian deterministic mutation
MU=2 Cauchy deterministic mutation
MU=3 Gaussian stochastic mutation
MU=4 Cauchy stochastic mutation
MU=99 (no mutation)

CH=1 Infeasibility discrimination
CH=2 Stochastic ranking
CH=3 Dynamic penalty

SE=1 Ranking selection
SE=2 Tournament selection
SE=3 Proportional selection

Table 6.2. Parameters of experimentation for typical run

Population size, n_{pop}	10
Epoch of termination, $epoch_{max}$	1000
Number of run	50
Dimension of function, n_{dim}	10 (eligible for f_{u1}, f_{u2}, f_{u3}, f_{u4}, f_{m1}, f_{m2}, f_{m3} and f_{m4})

Remark: In general, $epoch_{max}$ was 1000 in the experiment. However from the trial runs, the minimum $epoch_{max}$ for f_{c6} or f_{u4} was found to be 3000, while that for f_{m1} or f_{m2} to be 2000, in order to obtain satisfactory results.

Basically, the maximum number of tests for the constrained and unconstrained test functions was 1260 (= 7×4×5×3×3) and 540 (= 9×4×5×1×3) respectively. Due to the considerable number of combinations of EA operators for a variety of test functions, an extremely high computational cost would be needed to explore this problem exhaustively. Therefore, a series of preliminary tests was carried out to assess the opportunity for excluding any ineffective operators, so that the number of full runs could be reduced and useful results would be acquired from the empirical studies more efficiently. The detailed arrangements of the preliminary runs are discussed in Section 6.3 below.

6.3 Preliminary Runs

The primary objective of these tests was to exclude any EA operators which would have an unsatisfactory performance in handling the variety of test functions. In the following tests, fifty runs were used to obtain statistically significant results. The scope of these preliminary tests was to evaluate the following questions:

a. Are there any ineffective selection operators?

b. Are there any ineffective mutation operators?

c. Without mutation, is performance of EA still acceptable just with recombination?

6.3.1 Characterization of selection operators

A preliminary run was carried out for different selection operators, with Gaussian deterministic mutation, infeasibility discrimination constraint handling (MU=1 + CH=1) and no recombination (XO=99). The results are shown in Table 6.3, and the best among three combinations for each test function is highlighted in light grey.

Table 6.3. Preliminary run for different selection operators

	XO	99	99	99
	MU	1 (Gaussian)	1 (Gaussian)	1 (Gaussian)
	CH	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)
	SE	1 (Ranking)	2 (Tournament)	3 (Proportional)
	n_{pop}	10	10	10
	$epoch_{max}$	1000*	1000*	1000*
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions				
f_{c1} Floundas & P	-15	-14 (8.09×10^{-1})	-15 (1.98×10^{-1})	-12 (1.06)
f_{c2} Himmelblau 11	-30665.5	-30606.64 (105.50)	-30497.60 (157.42)	-30299.48 (314.76)
f_{c3} Floundas & P	-6961.81	-6933.65 (16.67)	-6942.47 (10.21)	-1227.26 (1494.92)
f_{c4} Hock & S 113	24.31	1115.35 (874.90)	872.59 (773.96)	1264.63 (1047.81)
f_{c5} Hock & S 100	680.63	681.08 (4.04×10^{-1})	681.56 (1.21)	2601840 (90.06)
f_{c6} Hock & S HX	7049.33	9508.65 (2402.18)	10340.99 (330.03)	14792.26 (5033.35)
f_{c7} Maa & Shanblatt	0.75	0.7480 (1.10×10^{-5})	0.7481 (4.53×10^{-4})	0.9359 (1.14×10^{-1})
Unconstrained unimodal functions				
f_{u1} Sphere Model	0	3.40×10^{-5} (1.32×10^{-5})	2.15×10^{-5} (6.03×10^{-6})	344.36 (780.72)
f_{u2} Schwefel 2.22	0	18.05 (17.34)	19.78 (16.39)	22.21 (13.94)
f_{u3} Schwefel 1.2	0	109.84 (279.63)	109.55 (233.36)	1879.31 (1426.64)
f_{u4} Rosenbrock	0	40.19 (102.42)	53.97 (154.35)	101.93 (234.80)
f_{u5} Easom	-1	-0.99996 (3.84×10^{-5})	-0.99998 (2.38×10^{-5})	-0.02811 (1.30×10^{-1})
Unconstrained multimodal functions				
f_{m1} Schwefel 7	-4189.83	-2443.34 (443.67)	-2570.82 (366.51)	-1750.35 (574.27)
f_{m2} Rastrigin	0	61.75 (18.49)	52.12 (13.34)	56.55 (15.82)
f_{m3} Ackley	0	14.92 (5.74)	18.24 (2.68)	18.28 (1.69)
f_{m4} Griewank	0	1.03×10^{-1} (5.07×10^{-2})	1.69×10^{-1} (1.64×10^{-1})	4.74 (5.63)

*Remark: For f_{c6}/f_{u4} and f_{m1}/f_{m2}, $epoch_{max}$ were 3000 and 2000 respectively.

From the results of the preliminary run for different selection operators in Table 6.3, there are a number of observations worthy of discussion as follows:

a. The highlighted results showed better performance with ranking selection (SE=1) or tournament selection (SE=2). Proportional selection (SE=3) could not provide the best result for any of the test functions.

b. For proportional selection, the results of two (f_{c3} and f_{c5}) out of seven constrained test functions and three (f_{u1}, f_{u3} and f_{u5}) out nine unconstrained test functions had unacceptable deviations from the respective solutions (as encompassed by the corrugated line type). However for ranking and tournament selection, their overall performance for all the test functions were acceptable, and there were no extreme situations like those found in the proportional selection.

c. The serious deviations of several constrained and unconstrained test functions when using proportional selection highlight the deficiencies of this selection operator. The main problem was relatively slow convergence rate or premature convergence, which reflected the epoch for evolution was insufficient. If the epoch of termination was increased in multiples, better results were achieved by the proportional selection. However it increased the expense of function evaluation, which would contradict the purpose of developing an effective EA with minimum function calls. In addition, according to Eq (5.22) in Section 5.3.2, the establishment of a probability vector for proportional selection would produce an ambiguous scenario whereby both positive and negative function values occurred in the same population during the search progress, so that the relative fitness would be confused. The choice of proportional selection would therefore hinder the search effectiveness for such kind of test functions.

Due to the limitations of proportional selection (SE=3) in the EA Suite for handling both constrained and unconstrained problems, it was not involved in the full runs. Only the ranking selection (SE=1) and tournament selection (SE=2) were used for the in-depth study.

6.3.2 Characterization of mutation operators

The second preliminary run was implemented to see whether there were significant differences between the performances of the mutation operators using a deterministic strategy parameter (MU=1 or 2) and a stochastic strategy parameter (MU=3 or 4). The combination of infeasibility constraint handling and ranking selection (CH=1 + SE=1) was used. The option of "no recombination" (XO=99) was included in order to assess the effect of the mutation operators more clearly. The results are shown in Table 6.4, with the best among the four combinations for each test function highlighted.

Table 6.4. Preliminary run for different mutation operators

	XO	99	99	99	99
	MU	1 (Gauss. Det.)	2 (Cau. Det.)	3 (Gauss. Sto.)	4 (Cau. Sto.)
	CH	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)
	SE	1 (Ranking)	1 (Ranking)	1 (Ranking)	1 (Ranking)
	n_{pop}	10	10	10	10
	$epoch_{max}$	1000*	1000*	1000*	1000*
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions					
f_{c1} Floundas & P	-15	-14 (8.09×10^{-1})	-13 (1.09)	-14 (8.81×10^{-1})	-13 (9.95×10^{-1})
f_{c2} Himmelblau 11	-30665.5	-30606.6 (105.50)	-30661.7 (17.42)	-30482.74 (202.17)	-30440.25 (374.5)
f_{c3} Floundas & P	-6961.81	-6933.65 (16.67)	-6910.59 (36.41)	-6475.56 (596.93)	-5785.93 (1282.80)
f_{c4} Hock & S 113	24.31	1115.35 (874.90)	26.37 (1.21)	3315.75 (1103.12)	3409.36 (1162.32)
f_{c5} Hock & S 100	680.63	681.08 (4.04×10^{-1})	680.91 (2.05×10^{-1})	403042.2 (1981247)	3329209 (4593692)
f_{c6} Hock & S HX	7049.33	9508.65 (2402.18)	7660.02 (360.00)	15707.44 (5725.57)	12225.74 (5485.85)
f_{c7} Maa & Shanblatt	0.75	0.748 (1.10×10^{-5})	0.748 (2.96×10^{-5})	0.8996 (1.10×10^{-1})	0.9945 (1.97×10^{-2})
Unconstrained unimodal functions					
f_{u1} Sphere Model	0	3.40×10^{-5} (1.32×10^{-5})	2.38×10^{-4} (9.04×10^{-5})	5262.14 (3283.48)	7755.64 (4412.40)
f_{u2} Schwefel 2.22	0	18.05 (17.34)	4.13×10^{-3} (1.04×10^{-3})	21.23 (8.60)	70.84 (76.93)
f_{u3} Schwefel 1.2	0	109.84 (279.63)	2.41×10^{-3} (2.20×10^{-3})	5615.81 (3343.71)	10328.69 (4702.32)
f_{u4} Rosenbrock	0	40.19 (102.42)	33.43 (60.42)	1085354 (1686403)	6386847 (9522267)
f_{u5} Easom	-1	-1.00 (3.84×10^{-5})	-1.00 (7.56×10^{-5})	-0.500464 (4.55×10^{-1})	-0.213780 (3.78×10^{-1})
Unconstrained multimodal functions					
f_{m1} Schwefel 7	-4189.83	-2443.34 (443.67)	-3688.29 (200.17)	-2495.23 (359.00)	-2499.82 (278.67)
f_{m2} Rastrigin	0	61.75 (18.49)	6.75 (2.88)	76.30 (20.93)	111.49 (18.09)
f_{m3} Ackley	0	14.92 (5.74)	7.16×10^{-3} (1.47×10^{-3})	13.53 (3.57)	18.54 (2.13)
f_{m4} Griewank	0	1.03×10^{-1} (5.07×10^{-2})	1.94×10^{-1} (9.97×10^{-2})	77.62 (39.14)	79.91 (44.40)

*Remark: For f_{c6}/f_{u4} and f_{m1}/f_{m2}, $epoch_{max}$ were 3000 and 2000 respectively.

The best results were obtained using the mutation operator of deterministic

strategy parameter with either Gaussian realization (MU=1) or Cauchy realization (MU=2). The mutation operators with stochastic strategy parameter (MU=3 and 4) could not provide the best result for any of the test functions. On the other hand, they gave unacceptable results for a number of test functions; f_{c4}, f_{c5}, f_{u1}, f_{u3} and f_{u4}. With both constrained and unconstrained functions, the major problem was premature convergence. As mentioned in Section 5.2.4, preventive measures had been applied to such mutation operators to prevent this happening, however this was ineffective for those five test functions. As a result, the mutation operators with the stochastic strategy parameters (MU=3 and 4) were excluded from the subsequent full runs.

6.3.3 Recombination without mutation

The third preliminary run was implemented to assess the performance of a recombination operator without mutation. This run was carried out by using different recombination operators together with infeasibility constraint handling and ranking selection (CH=1 + SE=1), as shown in Table 6.5. From this test run, excepting f_{c2} and f_{c7}, nearly all the results deviated significantly from the corresponding solutions. In the other words, the EA with recombination but no mutation could not determine the correct solutions. On the other hand, the performance of EA optimization was still satisfactory without the recombination as found in Section 6.3.2. Therefore it was concluded that although both recombination and mutation are type of variation, the role of mutation was crucial, but recombination had less importance. In the subsequent in-depth study, the use of both mutation and selection would be the foundation of EA, and the experiments designed to determine the robustness of different mutation and selection operators, as well as the appropriate use of the operators of recombination and constraint handling.

Table 6.5. Preliminary run for recombination without mutation

	XO	1 (Arithmetic)	2 (Uniform)	3 (Geometrical)
	MU	99	99	99
	CH	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)
	SE	1 (Ranking)	1 (Ranking)	1 (Ranking)
	n_{pop}	10	10	10
	$epoch_{max}$	1000*	1000*	1000*
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions				
f_{c1} Floundas & P	-15	-74.86 (27.27)	-54.60 (24.94)	-74.78 (28.47)
f_{c2} Himmelblau 11	-30665.5	-28256.84 (1101.71)	-28396.26 (1163.77)	-28383.22 (917.18)
f_{c3} Floundas & P	-6961.81	16397.98 (23345.45)	14597.54 (28084.42)	22630.98 (41473.16)
f_{c4} Hock & S 113	24.31	1699.07 (650.70)	1758.49 (903.88)	558.36 (158.91)
f_{c5} Hock & S 100	680.63	329818.4 (1012726)	1407101 (2567393)	1029282 (1800449)
f_{c6} Hock & S HX	7049.33	15431.56 (3817.31)	16269.92 (4990.77)	16591.51 (3280.55)
f_{c7} Maa & Shanblatt	0.75	0.9204 (1.23×10^{-1})	0.8819 (1.92×10^{-1})	0.7940 (8.28×10^{-2})
Unconstrained unimodal functions				
f_{u1} Sphere Model	0	5213.08 (1978.01)	13302.65 (4725.92)	18765.19 (5257.36)
f_{u2} Schwefel 2.22	0	25.53 (22.38)	516.12 (1088.03)	16508.18 (31369.51)
f_{u3} Schwefel 1.2	0	10290.51 (6579.95)	14408.08 (5914.94)	25993.18 (12339.92)
f_{u4} Rosenbrock	0	4759069 (4421671)	26006310 (16191740)	47480450 (19261420)
f_{u5} Easom	-1	-7.17×10^{-3} (5.07×10^{-2})	-6.23×10^{-33} (4.41×10^{-32})	-3.70×10^{-7} (2.43×10^{-6})
Unconstrained multimodal functions				
f_{m1} Schwefel 7	-4189.83	-1145.75 (373.10)	-1588.61 (415.95)	-1436.40 (370.91)
f_{m2} Rastrigin	0	78.68 (15.80)	106.88 (17.61)	121.38 (17.50)
f_{m3} Ackley	0	16.76 (1.71)	19.42 (7.02×10^{-1})	20.08 (0.58)
f_{m4} Griewank	0	53.34 (23.74)	122.33 (41.37)	175.69 (38.89)

*Remark: For f_{c6}/f_{u4} and f_{m1}/f_{m2}, $epoch_{max}$ were 3000 and 2000 respectively.

6.3.4 Summary of findings in preliminary runs

From the series of preliminary tests in Sections 6.3.1 to 6.3.3, proportional selection (SE=3) and the mutation operators with stochastic strategy parameter (MU=3 and 4) were excluded from the in-depth study, since their overall performance in different types of test functions was not satisfactory.

On the other hand, it was found that the role of mutation is crucial. For any choice of recombination operators, the global search may fail in the absence of a mutation operator. Without recombination, the search could still provide acceptable or satisfactory results. As a whole, this contrasts with the paradigm of genetic algorithm, which has a strong emphasis on recombination (crossover) but relatively little on mutation.

6.4 Summary

In this chapter, in order to evaluate the effectiveness of different EA operators described in Chapter 5, a variety of popular test functions in constrained or unconstrained; linear or nonlinear; unimodal or multimodal nature, were included in the experimentation. These test functions were chosen to be representative of different possible HVAC optimization problems in nature. By setting the combinations of the involved EA operators for empirical studies, it was found that the computational cost would be too high if the tests were exhaustive. Therefore a series of preliminary runs were implemented to study any ineffective selection and mutation operators, as well as the performance of EA without mutation. After the preliminary runs, the proportional selection and the mutation with stochastic strategy parameter were excluded due to their generally unsatisfactory performances in the test functions. The role of mutation in the

developed EA Suite was found to be very essential, even more important than the recombination. As a result, the EA operators to be carried forward for full runs in next chapter would be as follows:

a. Mutation operators
   - Gaussian deterministic mutation (MU=1)
   - Cauchy deterministic mutation (MU=2)

b. Selection operators
   - Ranking selection (SE=1)
   - Tournament selection (SE=2)

c. Recombination operators
   - Arithmetic recombination (XO=1)
   - Uniform recombination (XO=2)
   - Geometrical recombination (XO=3)

d. Constraint handling operators
   - Infeasibility discrimination (CH=1)
   - Stochastic ranking (CH=2)
   - Dynamic penalty (CH=3)

In next chapter, the full runs of these screened EA operators are described. The best combination of these available EA operators was identified, and a competent and robust EA formulated.

CHAPTER 7 FORMULATION OF ROBUST EA

In this chapter, a series of full runs was implemented for the screened EA operators from the preliminary runs in Chapter 6. In these full runs, each case was run fifty times. Then the mean, best and standard deviation were recorded for comparative study. In the full runs, the experiments were grouped into three main stages as follows:

1. Effective combination of mutation and selection operators.
2. Performance of global search at a reduced population n_{pop} and/or epoch of termination $epoch_{max}$.
3. Suitable choice of constraint handling operators.

In the first stage, the best combination of mutation and selection operators was identified, since they represent the core operators in the paradigms of evolutionary programming and evolution strategy. At the same time, the effectiveness of different recombination operators was evaluated. Based on the findings in the first stage, the search performances at the reduced n_{pop} and/or $epoch_{max}$ would be studied in the second stage, so that the robust combination of the operators of mutation, selection and recombination could be evaluated. Based on the findings in the second stage, the effectiveness of different constraint handling operators was studied in the third stage.

In this chapter, performance graphs of different test functions were used to illustrate the effectiveness of the related EA operators or their combinations. Each graph is the average performance of fifty runs, with the corresponding function value against epoch. The corresponding abbreviation is presented in one of the following formats:

"x□m□c□s□" for default n_{pop} and $epoch_{max}$;

"x□m□c□s□n#" for reduced n_{pop} and default $epoch_{max}$;

"x□m□c□s□e$" for default n_{pop} and reduced $epoch_{max}$; or

"x□m□c□s□n#e$" for both reduced n_{pop} and $epoch_{max}$.

where,

x: recombination

m: mutation

c: constraint handling

s: selection

n: population size

e: epoch of termination

□: option number of recombination, mutation, constraint handling or selection operator

#: reduced number of population, usually five

$: reduced number of epoch of termination, either three-quarters or half

Default n_{pop} and $epoch_{max}$ were 10 and 1000 respectively according to Table 6.2, however $epoch_{max}$ for test functions f_{c6} or f_{u4} was 3000, while that for f_{m1} or f_{m2} was 2000.

7.1 Choice of Mutation and Selection Operators

The first stage of the experiment was designed as a study focusing on the choice and contribution of mutation and selection operators. The possible combinations of the available mutation and selection operators, after screening from the preliminary runs, were as follows:

a. Gaussian deterministic mutation + Ranking selection [MU=1 + SE=1]

b. Gaussian deterministic mutation + Tournament selection [MU=1 + SE=2]

c. Cauchy deterministic mutation + Ranking selection [MU=2 + SE=1]

d. Cauchy deterministic mutation + Tournament selection [MU=2 + SE=2]

These combinations were run with the series of test functions to compare their performances of optimal search. The results are shown in Tables 7.1 to 7.4 as follows. The best result of each test function is highlighted in light grey.

Table 7.1. Performance based on Gaussian deterministic mutation and ranking selection

	XO	99	1 (Arithmetic)	2 (Uniform)	3 (Geometrical)
	MU	1 (Gaussian)	1 (Gaussian)	1 (Gaussian)	1 (Gaussian)
	CH	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)
	SE	1 (Ranking)	1 (Ranking)	1 (Ranking)	1 (Ranking)
	n_{pop}	10	10	10	10
	$epoch_{max}$	1000*	1000*	1000*	1000*
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions					
f_{c1} Floundas & P	-15	-14 (8.09×10^{-1})	-14 (8.10×10^{-1})	-14 (8.90×10^{-1})	-15 (4.52×10^{-1})
f_{c2} Himmelblau 11	-30665.5	-30606.64 (105.50)	-30571.45 (97.83)	-30576.76 (146.05)	-30532.40 (124.79)
f_{c3} Floundas & P	-6961.81	-6933.65 (16.67)	-6935.77 (14.23)	-6934.33 (15.80)	-6939.90 (11.70)
f_{c4} Hock & S 113	24.31	1115.35 (874.90)	921.23 (672.83)	588.84 (720.37)	45.48 (20.14)
f_{c5} Hock & S 100	680.63	681.08 (4.04×10^{-1})	681.21 (4.33×10^{-1})	680.94 (2.81×10^{-1})	685.18 (1.09)
f_{c6} Hock & S HX	7049.33	9508.65 (2402.18)	9024.35 (1858.95)	8923.25 (1670.58)	9119.64 (2118.32)
f_{c7} Maa & Shanblatt	0.75	0.7480 (1.10×10^{-5})	0.7480 (1.40×10^{-5})	0.7480 (1.39×10^{-5})	0.7480 (9.93×10^{-6})
Unconstrained unimodal functions					
f_{u1} Sphere Model	0	3.40×10^{-5} (1.32×10^{-5})	2.69×10^{-5} (7.66×10^{-6})	3.60×10^{-5} (1.12×10^{-5})	4.04×10^{-5} (1.27×10^{-5})
f_{u2} Schwefel 2.22	0	18.05 (17.34)	64.17 (1.91)	6.00 (12.68)	27.88 (17.06)
f_{u3} Schwefel 1.2	0	109.84 (279.63)	4.38×10^{-1} (2.43)	5.49 (18.60)	24690.73 (10320.74)
f_{u4} Rosenbrock	0	40.19 (102.42)	32.18 (65.04)	51.94 (107.08)	10.78 (5.85)
f_{u5} Easom	-1	-0.99996 (3.84×10^{-5})	-0.99997 (3.68×10^{-5})	-0.99997 (3.35×10^{-5})	-0.99997 (3.48×10^{-5})
Unconstrained multimodal functions					
f_{m1} Schwefel 7	-4189.83	-2443.34 (443.67)	-2190.56 (438.47)	-2594.99 (377.13)	-2972.36 (507.50)
f_{m2} Rastrigin	0	61.75 (18.49)	24.00 (11.25)	42.98 (16.90)	60.91 (11.68)
f_{m3} Ackley	0	14.92 (5.74)	3.40 (5.96)	9.90 (8.47)	18.59 (1.33)
f_{m4} Griewank	0	1.03×10^{-1} (5.07×10^{-2})	1.29×10^{-1} (5.65×10^{-2})	1.36×10^{-1} (7.39×10^{-2})	2.21×10^{-1} (1.58×10^{-1})

*Remark: For f_{c6}/f_{u4} and f_{m1}/f_{m2}, $epoch_{max}$ were 3000 and 2000 respectively.

Table 7.2. Performance based on Gaussian deterministic mutation and tournament selection

	XO	99	1 (Arithmetic)	2 (Uniform)	3 (Geometrical)
	MU	1 (Gaussian)	1 (Gaussian)	1 (Gaussian)	1 (Gaussian)
	CH	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	10	10	10	10
	$epoch_{max}$	1000*	1000*	1000*	1000*
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions					
f_{c1} Floundas & P	-15	-15 (1.98×10^{-1})	-15 (1.98×10^{-1})	-15 (1.41×10^{-1})	-15 (1.41×10^{-1})
f_{c2} Himmelblau 11	-30665.5	-30497.60 (157.42)	-30527.23 (146.92)	-30564.74 (112.88)	-30508.01 (160.63)
f_{c3} Floundas & P	-6961.81	-6942.47 (10.21)	-6940.62 (10.74)	-6943.18 (16.05)	-6940.18 (10.87)
f_{c4} Hock & S 113	24.31	872.59 (773.96)	746.29 (761.89)	441.29 (663.68)	42.28 (11.55)
f_{c5} Hock & S 100	680.63	681.56 (1.21)	681.06 (3.74×10^{-1})	681.27 (7.24×10^{-1})	1234.93 (2290.90)
f_{c6} Hock & S HX	7049.33	10340.99 (330.03)	10105.38 (2730.39)	10012.17 (3170.27)	9166.55 (2332.35)
f_{c7} Maa & Shanblatt	0.75	0.7481 (4.53×10^{-4})	0.7480 (1.31×10^{-5})	0.7480 (8.70×10^{-5})	0.7480 (1.57×10^{-4})
Unconstrained unimodal functions					
f_{u1} Sphere Model	0	2.15×10^{-5} (6.03×10^{-6})	2.10×10^{-5} (6.22×10^{-6})	2.02×10^{-5} (5.67×10^{-6})	2.16×10^{-5} (5.23×10^{-6})
f_{u2} Schwefel 2.22	0	19.78 (16.39)	3.99×10^{-1} (1.48)	2.14 (7.46)	29.16 (15.63)
f_{u3} Schwefel 1.2	0	109.55 (233.36)	1.81 (12.64)	6.24 (34.66)	27482.47 (11800.91)
f_{u4} Rosenbrock	0	53.97 (154.35)	15.62 (55.23)	62.17 (119.06)	11.79 (6.66)
f_{u5} Easom	-1	-0.99998 (2.38×10^{-5})	-0.99997 (2.54×10^{-5})	-0.99998 (2.44×10^{-5})	-0.97998 (1.41×10^{-1})
Unconstrained multimodal functions					
f_{m1} Schwefel 7	-4189.83	-2570.82 (366.51)	-2348.35 (339.42)	-2956.80 (355.24)	-2811.67 (449.74)
f_{m2} Rastrigin	0	52.12 (13.34)	23.04 (9.99)	35.20 (13.44)	60.53 (14.04)
f_{m3} Ackley	0	18.24 (2.68)	2.17 (5.06)	9.51 (8.46)	18.74 (5.11×10^{-1})
f_{m4} Griewank	0	1.69×10^{-1} (1.64×10^{-1})	1.78×10^{-1} (9.51×10^{-2})	1.75×10^{-1} (1.05)	2.61×10^{-1} (4.00×10^{-1})

*Remark: For f_{c6}/f_{u4} and f_{m1}/f_{m2}, $epoch_{max}$ were 3000 and 2000 respectively.

Table 7.3. Performance based on Cauchy deterministic mutation and ranking selection

	XO	99	1 (Arithmetic)	2 (Uniform)	3 (Geometrical)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)
	SE	1 (Ranking)	1 (Ranking)	1 (Ranking)	1 (Ranking)
	n_{pop}	10	10	10	10
	$epoch_{max}$	1000*	1000*	1000*	1000*
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions					
f_{c1} Floundas & P	-15	-13 (1.09)	-13 (9.26×10^{-1})	-13 (9.37×10^{-1})	-14 (9.29×10^{-1})
f_{c2} Himmelblau 11	-30665.5	-30661.70 (17.42)	-30646.15 (67.93)	-30659.63 (24.84)	-30665.06 (5.72)
f_{c3} Floundas & P	-6961.81	-6910.59 (36.41)	-6909.86 (27.56)	-6907.01 (32.12)	-6914.55 (28.17)
f_{c4} Hock & S 113	24.31	26.37 (1.21)	25.81 (1.28)	25.96 (1.11)	26.30 (1.44)
f_{c5} Hock & S 100	680.63	680.91 (2.05×10^{-1})	680.90 (1.97×10^{-1})	680.87 (1.72×10^{-1})	685.43 (1.70)
f_{c6} Hock & S HX	7049.33	7660.02 (360.00)	7528.54 (400.87)	7726.59 (468.75)	7481.92 (348.81)
f_{c7} Maa & Shanblatt	0.75	0.7480 (2.96×10^{-5})	0.7480 (4.66×10^{-5})	0.7480 (6.06×10^{-5})	0.7481 (1.33×10^{-4})
Unconstrained unimodal functions					
f_{u1} Sphere Model	0	2.38×10^{-4} (9.04×10^{-5})	2.23×10^{-4} (8.79×10^{-5})	3.19×10^{-4} (1.09×10^{-4})	4.84×10^{-4} (2.18×10^{-4})
f_{u2} Schwefel 2.22	0	4.13×10^{-3} (1.04×10^{-3})	3.69×10^{-3} (7.98×10^{-4})	4.06×10^{-3} (7.61×10^{-4})	2.03 (10.12)
f_{u3} Schwefel 1.2	0	2.41×10^{-3} (2.20×10^{-3})	1.36×10^{-3} (9.29×10^{-4})	2.05×10^{-3} (1.26×10^{-3})	27977.86 (14018.18)
f_{u4} Rosenbrock	0	33.43 (60.42)	33.95 (98.28)	42.79 (91.85)	13.19 (6.59)
f_{u5} Easom	-1	-0.99990 (7.56×10^{-5})	-0.99992 (9.72×10^{-5})	-0.99988 (1.20×10^{-4})	-0.99990 (8.73×10^{-5})
Unconstrained multimodal functions					
f_{m1} Schwefel 7	-4189.83	-3688.29 (200.17)	-3711.25 (197.42)	-3752.12 (203.19)	-4189.83 (6.55×10^{-4})
f_{m2} Rastrigin	0	6.75 (2.88)	4.77 (1.96)	5.61 (2.40)	7.14 (3.07)
f_{m3} Ackley	0	7.16×10^{-3} (1.47×10^{-3})	5.97×10^{-3} (1.44×10^{-3})	4.04×10^{-1} (2.80)	1.26 (4.68)
f_{m4} Griewank	0	1.94×10^{-1} (9.97×10^{-2})	1.82×10^{-1} (1.01×10^{-1})	1.78×10^{-1} (7.41×10^{-2})	2.51×10^{-1} (1.03×10^{-1})

*Remark: For f_{c6}/f_{u4} and f_{m1}/f_{m2}, $epoch_{max}$ were 3000 and 2000 respectively.

Table 7.4. Performance based on Cauchy deterministic mutation and tournament selection

	XO	99	1 (Arithmetic)	2 (Uniform)	3 (Geometrical)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	10	10	10	10
	$epoch_{max}$	1000*	1000*	1000*	1000*
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions					
f_{c1} Floundas & P	-15	-14 (7.41×10^{-1})	-14 (8.02×10^{-1})	-14 (7.94×10^{-1})	-14 (8.03×10^{-1})
f_{c2} Himmelblau 11	-30665.5	-30646 (69.60)	-30647.53 (52.83)	-30658.2 (25.57)	-30640.65 (63.43)
f_{c3} Floundas & P	-6961.81	-6930.84 (19.55)	-6929.11 (17.61)	-6931.37 (14.71)	-6927.57 (17.11)
f_{c4} Hock & S 113	24.31	26.32 (1.14)	25.89 (9.90×10^{-1})	26.23 (1.42)	25.89 (9.54×10^{-1})
f_{c5} Hock & S 100	680.63	680.84 (1.11×10^{-1})	680.85 (1.34×10^{-1})	680.84 (1.17×10^{-1})	685.79 (2.41)
f_{c6} Hock & S HX	7049.33	7757.07 (873.51)	7611.68 (638.52)	7964.68 (967.03)	8079.18 (1361.11)
f_{c7} Maa & Shanblatt	0.75	0.7484 (9.79×10^{-4})	0.7487 (1.84×10^{-3})	0.7484 (1.40×10^{-3})	0.7492 (2.91×10^{-3})
Unconstrained unimodal functions					
f_{u1} Sphere Model	0	8.59×10^{-5} (2.97×10^{-5})	7.48×10^{-5} (2.62×10^{-5})	7.73×10^{-5} (2.77×10^{-5})	8.54×10^{-5} (3.18×10^{-5})
f_{u2} Schwefel 2.22	0	2.23×10^{-3} (4.41×10^{-4})	2.19×10^{-3} (4.64×10^{-4})	2.20×10^{-3} (4.33×10^{-4})	2.34×10^{-3} (4.35×10^{-4})
f_{u3} Schwefel 1.2	0	4.35×10^{-4} (2.49×10^{-4})	3.65×10^{-4} (2.21×10^{-4})	3.83×10^{-4} (2.46×10^{-4})	29989.42 (14308.52)
f_{u4} Rosenbrock	0	45.38 (108.32)	7.90 (14.81)	14.50 (62.25)	12.54 (6.49)
f_{u5} Easom	-1	-0.99996 (4.78×10^{-5})	-0.99997 (3.08×10^{-5})	-0.99995 (4.38×10^{-5})	-0.99997 (3.09×10^{-5})
Unconstrained multimodal functions					
f_{m1} Schwefel 7	-4189.83	-3704.17 (184.18)	-3725.95 (181.19)	-3820.24 (170.36)	-4135.35 (385.24)
f_{m2} Rastrigin	0	3.34 (1.81)	3.23 (1.98)	3.24 (1.38)	4.14 (2.07)
f_{m3} Ackley	0	0.64 (3.26)	3.48×10^{-3} (6.41×10^{-4})	3.71×10^{-3} (7.16×10^{-4})	2.66 (6.66)
f_{m4} Griewank	0	1.89×10^{-1} (7.59×10^{-2})	1.85×10^{-1} (1.17×10^{-1})	1.88×10^{-1} (8.12×10^{-2})	1.85×10^{-1} (1.09×10^{-1})

*Remark: For f_{c6}/f_{u4} and f_{m1}/f_{m2}, $epoch_{max}$ were 3000 and 2000 respectively.

7.1.1 Gaussian deterministic mutation

From Tables 7.1 to 7.4, generally the combinations of [MU=1 + SE=1] and [MU=1 + SE=2] had a less satisfactory performance as compared to the other two combinations. The test function f_{c4} (Hock and Schittkowski's Problem 113) had problem of effectiveness with Gaussian deterministic mutation, and its graph is used to illustrate this observation, as shown in Fig. 7.1. Most of the combinations [MU=1 + SE=1] and [MU=1 + SE=2] deviated significantly from the solution.

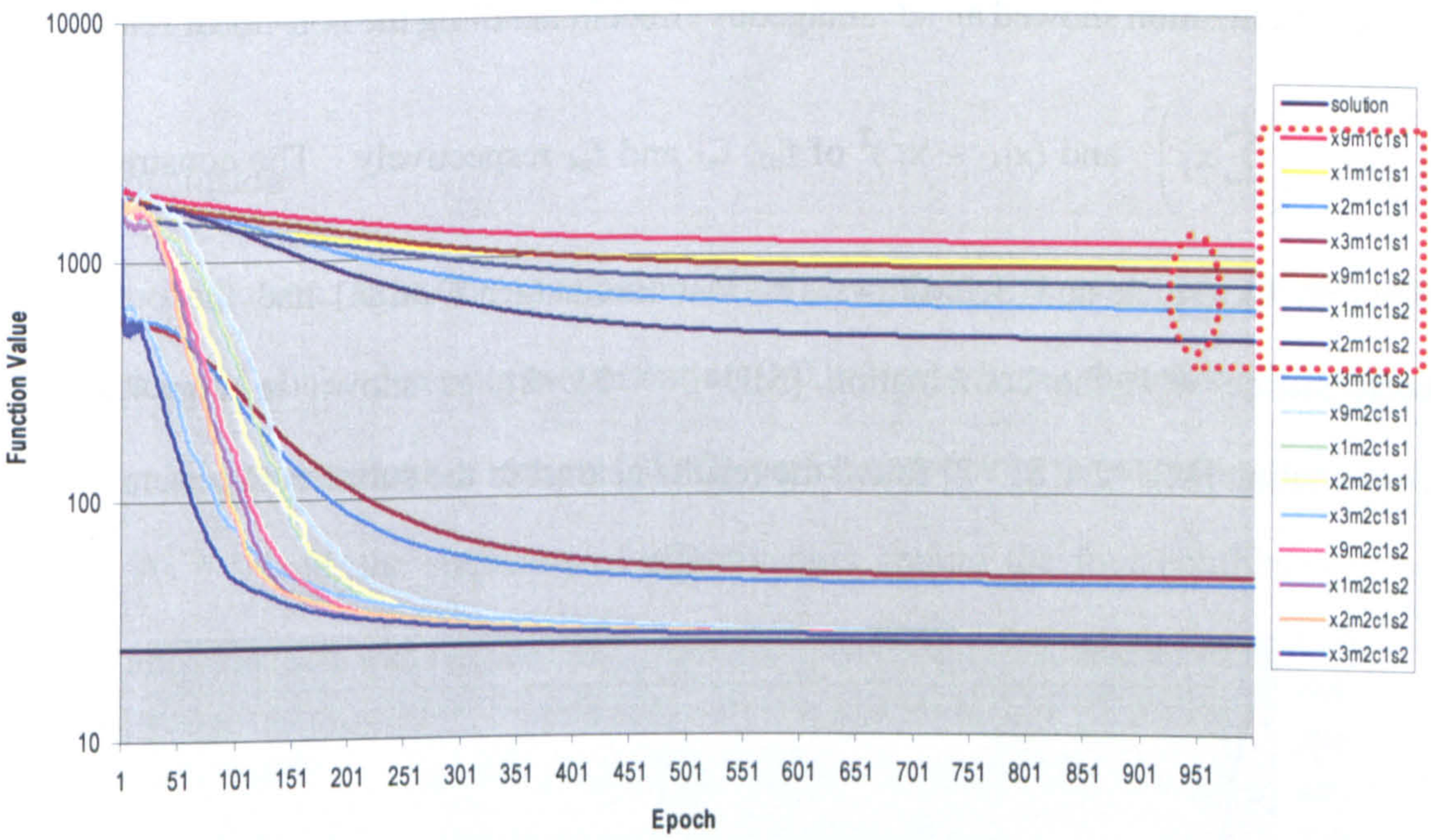


Fig. 7.1. Performance graphs for constrained test function f_{c4}.

7.1.2 Effectiveness of Cauchy deterministic mutation

For the test functions f_{u2} (Schwefel's Problem 2.22), f_{u3} (Schwefel's Problem 1.2) and f_{u4} (Generalized Rosenbrock's Function), the combination [MU=2 + SE=2] gave the best performance, and Cauchy realization (MU=2) had a better performance than Gaussian realization (MU=1). The test functions f_{u2}, f_{u3} and f_{u4} are expressed as follows

(refer to Eqs (A14), (A15) and (A16) respectively in Appendix I for detail.)

$$f_{u2}(x) = \sum_{i=1}^{10}|x_i| + \prod_{i=1}^{10}|x_i| \tag{7.1}$$

$$f_{u3}(x) = \sum_{i=1}^{10}\left(\sum_{j=1}^{i} x_j\right)^2 \tag{7.2}$$

$$f_{u4}(x) = \sum_{i=1}^{9}[100(x_{i+1} - x_i^{\,2})^2 + (x_i - 1)^2] \tag{7.3}$$

Cauchy realization showed an advantageous effect in handling the non-linear relationship $\prod_{i=1}^{10}|x_i|$, $\left(\sum_{j=1}^{i} x_j\right)^2$ and $(x_{i+1} - x_i^2)^2$ of f_{u2}, f_{u3} and f_{u4} respectively. The constrained test function f_{c6} (Hock and Schittkowski's Heat Exchanger Design) had the outstanding performance with the combination [MU=2 + SE=2], as shown in Fig. 7.2. The combination [MU=2 + SE=2] found the results nearest to the solution in general.

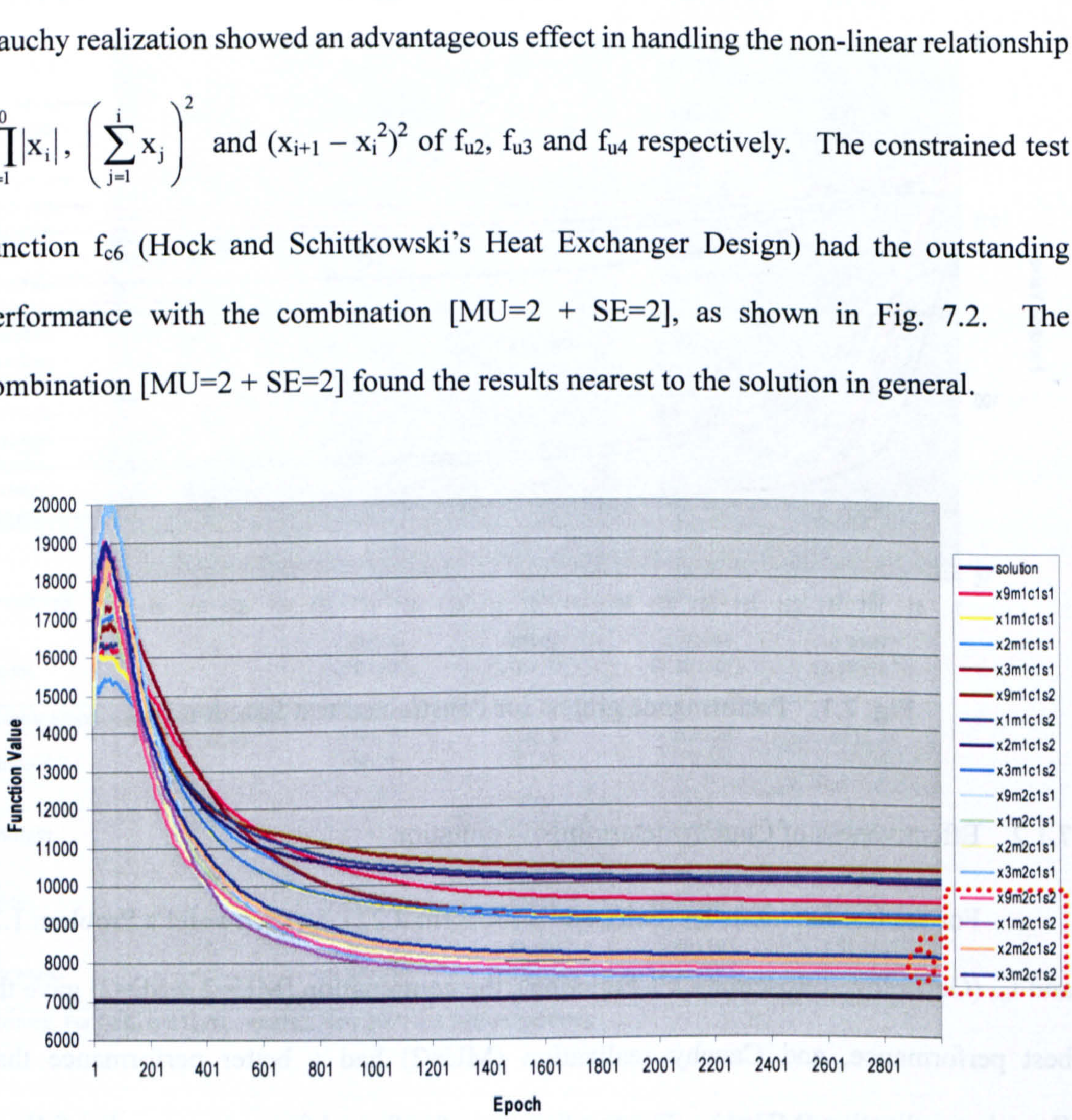


Fig. 7.2. Performance graphs for constrained test function f_{c6}.

7.1.3 Applicability of Gaussian deterministic mutation

For f_{u1} (Sphere Model), the combination of [MU=1 + SE=2] gave the best performance, and Gaussian realization (MU=1) was better than Cauchy realization (MU=2) in this case. This was because the Gaussian random number provided a smaller mutation step close to the optimum of the unimodal landscape of f_{u1}, and the effects of random perturbation were accordingly less. However, the Cauchy random number had a greater perturbation, which was more effective in handling multimodal problems and in preventing the search from being trapped at local optima.

7.1.4 Conclusion

In conclusion, from the viewpoint of either mutation or selection, the general performance of Cauchy deterministic mutation (MU=2) was better than that of Gaussian (MU=1), while tournament selection (SE=2) performed better than ranking selection (SE=1). As a whole, the sequence of effectiveness among the four combinations of mutation and selection was found to be:

[MU=2 + SE=2] >> [MU=2 + SE=1] >> [MU=1 + SE=2] >> [MU=1 + SE=1]

where ">>" implies "was more effective than".

The combination of [MU=2 + SE=2] generally gave the best results for different problem types (constrained, unconstrained unimodal and unconstrained multimodal) of the test functions. Based on the promising performance of the combination [MU=2 + SE=2], further in-depth analysis was carried out to test its association with different recombination operators. A series of more stringent situations, such as reduced population and/or epoch, were applied to the combination of [MU=2 + SE=2] so that the

potential improvement using recombination could be identified. This is more fully discussed in Section 7.3.

7.2 Role and Choice of Recombination Operators

7.2.1 Contribution of recombination

Within the regime of EA, evolutionary programming uses mutation as the essential operator for global searching, while evolution strategy involves both recombination and mutation. From Tables 7.5 to 7.9, together with Tables 7.1 to 7.4 before, the best results in each row were generally found with the involvement of the recombination operator, though there were a few exceptions.

For the combination of [MU=2 + SE=2], at a reduced population and/or epoch (Tables 7.5 to 7.9), the results without the involvement of recombination were less satisfactory. So the existence of a suitable option of recombination could improve the chance that the optimum or near-optimum could be determined, even though the population and epoch might be limited. This was especially important for optimization problems developed through the plant simulation model, which demands considerable computational resources. In other words, the paradigm of evolution strategy should produce better and more reliable results than evolutionary programming.

Table 7.5. Performance at half population

	XO	99	1 (Arithmetic)	2 (Uniform)	3 (Geometrical)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	5	5	5	5
	$epoch_{max}$	1000*	1000*	1000*	1000*
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions					
f_{c1} Floundas & P	-15	-13 (1.05)	-13 (9.98×10^{-1})	-13 (1.07)	-13 (1.14)
f_{c2} Himmelblau 11	-30665.5	-30588.29 (114.81)	-30567.74 (111.14)	-30552.03 (168.33)	-30600.94 (103.87)
f_{c3} Floundas & P	-6961.81	-6855.25 (68.04)	-6845.92 (72.92)	-6845.45 (57.34)	-6846.05 (76.34)
f_{c4} Hock & S 113	24.31	27.39 (7.62)	25.47 (8.84)	26.92 (8.90)	27.07 (7.71)
f_{c5} Hock & S 100	680.63	668.09 (96.43)	681.62 (9.20×10^{-1})	654.30 (134.92)	672.96 (97.15)
f_{c6} Hock & S HX	7049.33	8005.74 (3215.60)	8514.78 (3452.18)	8750.25 (3181.40)	8818.18 (2180.84)
f_{c7} Maa & Shanblatt	0.75	0.7984 (3.77×10^{-2})	0.8037 (3.72×10^{-2})	0.7954 (3.92×10^{-2})	0.7992 (3.90×10^{-2})
Unconstrained unimodal functions					
f_{u1} Sphere Model	0	1.68×10^{-4} (8.17×10^{-5})	1.56×10^{-4} (7.18×10^{-5})	1.90×10^{-4} (9.17×10^{-5})	1.78×10^{-4} (9.64×10^{-5})
f_{u2} Schwefel 2.22	0	5.65×10^{-3} (1.60×10^{-2})	3.29×10^{-3} (8.58×10^{-4})	3.44×10^{-3} (6.62×10^{-4})	9.66×10^{-1} (6.52)
f_{u3} Schwefel 1.2	0	2.07×10^{-1} (5.40×10^{-1})	2.29×10^{-1} (7.26×10^{-1})	1.49 (6.44)	49436.57 (29528.78)
f_{u4} Rosenbrock	0	32.31 (83.69)	39.43 (83.82)	18.74 (46.59)	12.05 (7.03)
f_{u5} Easom	-1	-0.89988 (0.303)	-0.95988 (1.98×10^{-1})	-0.95987 (1.98×10^{-1})	-0.95987 (1.98×10^{-1})
Unconstrained multimodal functions					
f_{m1} Schwefel 7	-4189.83	-3531.04 (237.27)	-3495.63 (248.23)	-3481.42 (273.40)	-3933.47 (567.31)
f_{m2} Rastrigin	0	12.46 (5.84)	9.55 (5.08)	10.96 (5.00)	11.09 (5.44)
f_{m3} Ackley	0	5.29 (8.58)	5.50×10^{-3} (1.32×10^{-3})	2.59 (6.38)	6.58 (9.02)
f_{m4} Griewank	0	1.65×10^{-1} (9.94×10^{-2})	1.53×10^{-1} (1.01×10^{-1})	1.77×10^{-1} (1.25×10^{-1})	1.83×10^{-1} (1.32×10^{-1})

*Remark: For f_{c6}/f_{u4} and f_{m1}/f_{m2}, $epoch_{max}$ were 3000 and 2000 respectively.

Table 7.6. Performance at three-quarters epoch

	XO	99	1 (Arithmetic)	2 (Uniform)	3 (Geometrical)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	10	10	10	10
	$epoch_{max}$	750*	750*	750*	750*
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions					
f_{c1} Floundas & P	-15	-14 (9.04×10^{-1})	-14 (8.86×10^{-1})	-14 (9.34×10^{-1})	-14 (7.67×10^{-1})
f_{c2} Himmelblau 11	-30665.5	-30638.9 (60.04)	-30648.12 (53.71)	-30644.10 (59.21)	-30648.30 (51.63)
f_{c3} Floundas & P	-6961.81	-6921.03 (19.57)	-6919.34 (23.88)	-6925.83 (19.40)	-6921.74 (20.37)
f_{c4} Hock & S 113	24.31	26.66 (1.49)	26.47 (1.27)	26.59 (1.51)	26.86 (1.87)
f_{c5} Hock & S 100	680.63	680.96 (2.38×10^{-1})	680.94 (1.93×10^{-1})	681.00 (2.78×10^{-1})	686.85 (2.65)
f_{c6} Hock & S HX	7049.33	8073.32 (1208.58)	7813.88 (794.98)	8236.42 (1362.01)	7738.23 (1057.43)
f_{c7} Maa & Shanblatt	0.75	0.7522 (9.92×10^{-3})	0.7528 (1.18×10^{-2})	0.7517 (8.46×10^{-3})	0.7520 (1.18×10^{-2})
Unconstrained unimodal functions					
f_{u1} Sphere Model	0	1.05×10^{-4} (3.68×10^{-5})	9.34×10^{-5} (3.35×10^{-5})	1.00×10^{-4} (4.04×10^{-5})	1.02×10^{-4} (3.55×10^{-5})
f_{u2} Schwefel 2.22	0	2.48×10^{-3} (5.81×10^{-4})	2.42×10^{-3} (4.86×10^{-4})	2.46×10^{-3} (4.97×10^{-4})	2.07×10^{-1} (1.44)
f_{u3} Schwefel 1.2	0	9.32×10^{-3} (3.05×10^{-2})	9.68×10^{-4} (9.65×10^{-4})	1.21×10^{-3} (1.29×10^{-3})	25777.88 (11446.59)
f_{u4} Rosenbrock	0	71.44 (137.36)	25.57 (62.44)	43.21 (87.82)	13.64 (5.51)
f_{u5} Easom	-1	-0.99995 (4.40×10^{-5})	-0.99996 (4.99×10^{-5})	-0.99996 (4.37×10^{-5})	-0.97995 (1.41×10^{-1})
Unconstrained multimodal functions					
f_{m1} Schwefel 7	-4189.83	-3742.44 (196.99)	-3740.52 (189.62)	-3774.04 (159.44)	-4151.81 (161.57)
f_{m2} Rastrigin	0	4.66 (2.07)	4.12 (2.30)	5.57 (2.17)	6.71 (3.12)
f_{m3} Ackley	0	3.99×10^{-1} (2.79)	3.76×10^{-3} (7.13×10^{-4})	3.95×10^{-3} (7.69×10^{-4})	2.74 (6.86)
f_{m4} Griewank	0	1.72×10^{-1} (7.80×10^{-2})	1.77×10^{-1} (8.88×10^{-2})	1.99×10^{-1} (1.06×10^{-1})	1.65×10^{-1} (1.10×10^{-1})

*Remark: For f_{c6} and $f_{u4}/f_{m1}/f_{m2}$, $epoch_{max}$ were 2250 and 1500 respectively.

Table 7.7. Performance at half epoch

	XO	99	1 (Arithmetic)	2 (Uniform)	3 (Geometrical)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	10	10	10	10
	$epoch_{max}$	500*	500*	500*	500*
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions					
f_{c1} Floundas & P	-15	-14 (9.57×10^{-1})	-14 (1.09)	-14 (1.00)	-14 (8.04×10^{-1})
f_{c2} Himmelblau 11	-30665.5	-30612.91 (113.12)	-30603.78 (112.82)	-30625.74 (94.66)	-30624.02 (90.42)
f_{c3} Floundas & P	-6961.81	-6899.03 (30.93)	-6904.97 (25.89)	-6898.01 (31.32)	-6906.90 (26.56)
f_{c4} Hock & S 113	24.31	29.23 (4.16)	30.51 (14.14)	27.78 (2.66)	28.41 (2.70)
f_{c5} Hock & S 100	680.63	681.31 (6.25×10^{-1})	681.13 (32.40)	681.22 (3.38×10^{-1})	686.23 (2.27)
f_{c6} Hock & S HX	7049.33	8216.48 (1117.71)	8275.74 (2015.17)	8459.77 (1311.58)	7983.55 (1322.23)
f_{c7} Maa & Shanblatt	0.75	0.7694 (2.28×10^{-2})	0.7718 (2.68×10^{-2})	0.7662 (2.42×10^{-2})	0.7713 (2.66×10^{-2})
Unconstrained unimodal functions					
f_{u1} Sphere Model	0	1.46×10^{-4} (5.05×10^{-5})	1.15×10^{-4} (3.09×10^{-5})	1.48×10^{-4} (5.45×10^{-5})	1.55×10^{-4} (5.48×10^{-5})
f_{u2} Schwefel 2.22	0	2.97×10^{-3} (7.58×10^{-4})	3.00×10^{-3} (5.67×10^{-4})	2.90×10^{-3} (6.19×10^{-4})	8.18×10^{-2} (5.56×10^{-1})
f_{u3} Schwefel 1.2	0	16.17 (31.72)	8.49×10^{-1} (1.43)	11.76 (20.01)	31460.80 (15460.22)
f_{u4} Rosenbrock	0	64.56 (104.49)	45.40 (109.76)	58.46 (104.02)	14.72 (5.04)
f_{u5} Easom	-1	-0.95995 (1.98×10^{-1})	-0.99993 (6.48×10^{-5})	-0.97993 (1.41×10^{-1})	-0.87994 (3.28×10^{-1})
Unconstrained multimodal functions					
f_{m1} Schwefel 7	-4189.83	-3613.59 (220.29)	-3591.05 (216.43)	-3705.93 (223.98)	-4052.84 (507.74)
f_{m2} Rastrigin	0	6.73 (3.33)	5.93 (2.51)	7.04 (3.28)	9.61 (4.06)
f_{m3} Ackley	0	1.84 (5.55)	4.54×10^{-3} (8.55×10^{-4})	4.74×10^{-3} (8.76×10^{-4})	2.95 (6.62)
f_{m4} Griewank	0	1.75×10^{-1} (9.68×10^{-2})	1.71×10^{-1} (8.79×10^{-2})	1.66×10^{-1} (1.10×10^{-1})	1.98×10^{-1} (1.01×10^{-1})

*Remark: For f_{c6} and $f_{u4}/f_{m1}/f_{m2}$, $epoch_{max}$ were 1500 and 1000 respectively.

Table 7.8. Performance at half population and three-quarters epoch

	XO	99	1 (Arithmetic)	2 (Uniform)	3 (Geometrical)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	5	5	5	5
	$epoch_{max}$	750*	750*	750*	750*
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions					
f_{c1} Floundas & P	-15	-13 (1.03)	-13 (9.29×10^{-1})	-12 (1.20)	-13 (1.21)
f_{c2} Himmelblau 11	-30665.5	-30553.53 (137.15)	-30561.93 (130.38)	-30533.26 (183.15)	-30580.51 (154.66)
f_{c3} Floundas & P	-6961.81	-6530.24 (1093.38)	-6771.17 (124.96)	-6788.75 (119.72)	-6799.61 (110.47)
f_{c4} Hock & S 113	24.31	46.14 (100.03)	32.10 (10.90)	30.40 (7.85)	28.25 (6.72)
f_{c5} Hock & S 100	680.63	641.48 (163.72)	668.63 (96.49)	669.12 (96.59)	674.94 (97.52)
f_{c6} Hock & S HX	7049.33	9340.00 (3582.73)	9386.44 (4271.41)	9514.46 (4022.58)	8057.86 (4628.77)
f_{c7} Maa & Shanblatt	0.75	0.81498 (1.27×10^{-1})	0.8047 (1.26×10^{-1})	0.8047 (1.73×10^{-1})	0.8027 (1.24×10^{-1})
Unconstrained unimodal functions					
f_{u1} Sphere Model	0	2.08×10^{-4} (1.23×10^{-4})	2.22×10^{-4} (1.19×10^{-4})	1.91×10^{-4} (9.35×10^{-5})	2.25×10^{-4} (1.14×10^{-4})
f_{u2} Schwefel 2.22	0	3.21 (12.85)	6.86×10^{-1} (4.83)	8.89×10^{-1} (6.26)	3.63 (12.53)
f_{u3} Schwefel 1.2	0	27.51 (70.28)	10.07 (20.76)	19.61 (27.83)	43726.20 (28554.57)
f_{u4} Rosenbrock	0	150.15 (298.89)	77.00 (190.88)	117.75 (262.09)	12.24 (7.03)
f_{u5} Easom	-1	-0.85987 (3.50×10^{-1})	-0.93988 (2.40×10^{-1})	-0.95985 (1.98×10^{-1})	-0.87987 (3.28×10^{-1})
Unconstrained multimodal functions					
f_{m1} Schwefel 7	-4189.83	-3461.77 (271.07)	-3433.20 (266.15)	-3412.76 (248.63)	-3885.27 (541.72)
f_{m2} Rastrigin	0	13.13 (7.36)	11.26 (4.85)	14.17 (6.71)	16.61 (9.31)
f_{m3} Ackley	0	6.94 (9.00)	7.88×10^{-2} (3.63×10^{-1})	3.66 (7.31)	5.68 (8.08)
f_{m4} Griewank	0	1.90×10^{-1} (1.10×10^{-1})	1.37×10^{-1} (6.93×10^{-2})	1.67×10^{-1} (1.01×10^{-1})	1.77×10^{-1} (8.33×10^{-2})

*Remark: For f_{c6} and $f_{u4}/f_{m1}/f_{m2}$, $epoch_{max}$ were 2250 and 1500 respectively.

Table 7.9. Performance at half population and half epoch

	XO	99	1 (Arithmetic)	2 (Uniform)	3 (Geometrical)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)	1 (Infeasibility)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	5	5	5	5
	$epoch_{max}$	500*	500*	500*	500*
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions					
f_{c1} Floundas & P	-15	-12 (1.08)	-12 (1.04)	-12 (1.16)	-12 (1.14)
f_{c2} Himmelblau 11	-30665.5	-30449.17 (223.66)	-30378.70 (214.04)	-30432.03 (258.38)	-30386.06 (273.57)
f_{c3} Floundas & P	-6961.81	-6519.63 (572.53)	-6560.40 (376.55)	-6499.30 (427.92)	-6472.44 (792.71)
f_{c4} Hock & S 113	24.31	85.59 (197.30)	129.59 (292.13)	90.52 (173.01)	31.14 (12.82)
f_{c5} Hock & S 100	680.63	672.93 (97.30)	684.72 (4.74)	684.99 (3.38)	686.19 (7.74)
f_{c6} Hock & S HX	7049.33	9696.39 (4130.56)	8770.05 (3991.54)	10004.72 (4180.32)	9549.48 (3857.36)
f_{c7} Maa & Shanblatt	0.75	0.8605 (1.35×10^{-1})	0.8699 (5.42×10^{-2})	0.8741 (5.55×10^{-2})	0.8761 (4.95×10^{-2})
Unconstrained unimodal functions					
f_{u1} Sphere Model	0	4.47×10^{-4} (2.11×10^{-4})	3.24×10^{-4} (1.42×10^{-4})	3.60×10^{-4} (1.76×10^{-4})	3.74×10^{-4} (1.88×10^{-4})
f_{u2} Schwefel 2.22	0	2.16 (8.67)	4.54×10^{-3} (1.22×10^{-3})	1.07 (5.23)	5.92 (13.37)
f_{u3} Schwefel 1.2	0	339.07 (309.91)	140.17 (204.42)	298.42 (261.69)	35592.54 (19960.57)
f_{u4} Rosenbrock	0	142.70 (278.51)	140.53 (324.68)	154.01 (386.79)	13.98 (6.80)
f_{u5} Easom	-1	-0.89974 (3.03×10^{-1})	-0.91981 (2.74×10^{-1})	-0.87982 (3.28×10^{-1})	-0.93983 (2.40×10^{-1})
Unconstrained multimodal functions					
f_{m1} Schwefel 7	-4189.83	-3340.17 (251.36)	-3318.97 (296.32)	-3306.03 (303.80)	-3824.34 (302.70)
f_{m2} Rastrigin	0	17.55 (8.97)	15.34 (8.26)	19.42 (9.38)	17.72 (8.33)
f_{m3} Ackley	0	5.11 (7.85)	6.67×10^{-1} (2.59)	3.93 (7.13)	11.33 (8.53)
f_{m4} Griewank	0	1.86×10^{-1} (1.12×10^{-1})	1.86×10^{-1} (1.27×10^{-1})	1.91×10^{-1} (1.19×10^{-1})	1.90×10^{-1} (1.05×10^{-1})

*Remark: For f_{c6} and $f_{u4}/f_{m1}/f_{m2}$, $epoch_{max}$ were 1500 and 1000 respectively.

7.2.2 Effectiveness of arithmetic recombination

The combinations of [MU=1 + SE=1], [MU=1 + SE=2] and [MU=2 + SE=1] associating with arithmetic recombination (XO=1) gave the best results with the unconstrained test functions, while geometrical recombination (XO=3) worked best for the constrained test functions. For the combination of [MU=2 + SE=2], arithmetic recombination (XO=1) also produced the best results for the unconstrained test functions, but the performance of the different recombination operators was comparable for the constrained test functions, as shown in Table 7.4. A graph of test function f_{u2} (Schwefel's Problem 2.22) typically illustrates the outstanding performance of arithmetic recombination (XO=1) in association with the combination of [MU=2 + SE=2] for the unconstrained test functions, is shown in Fig. 7.3.

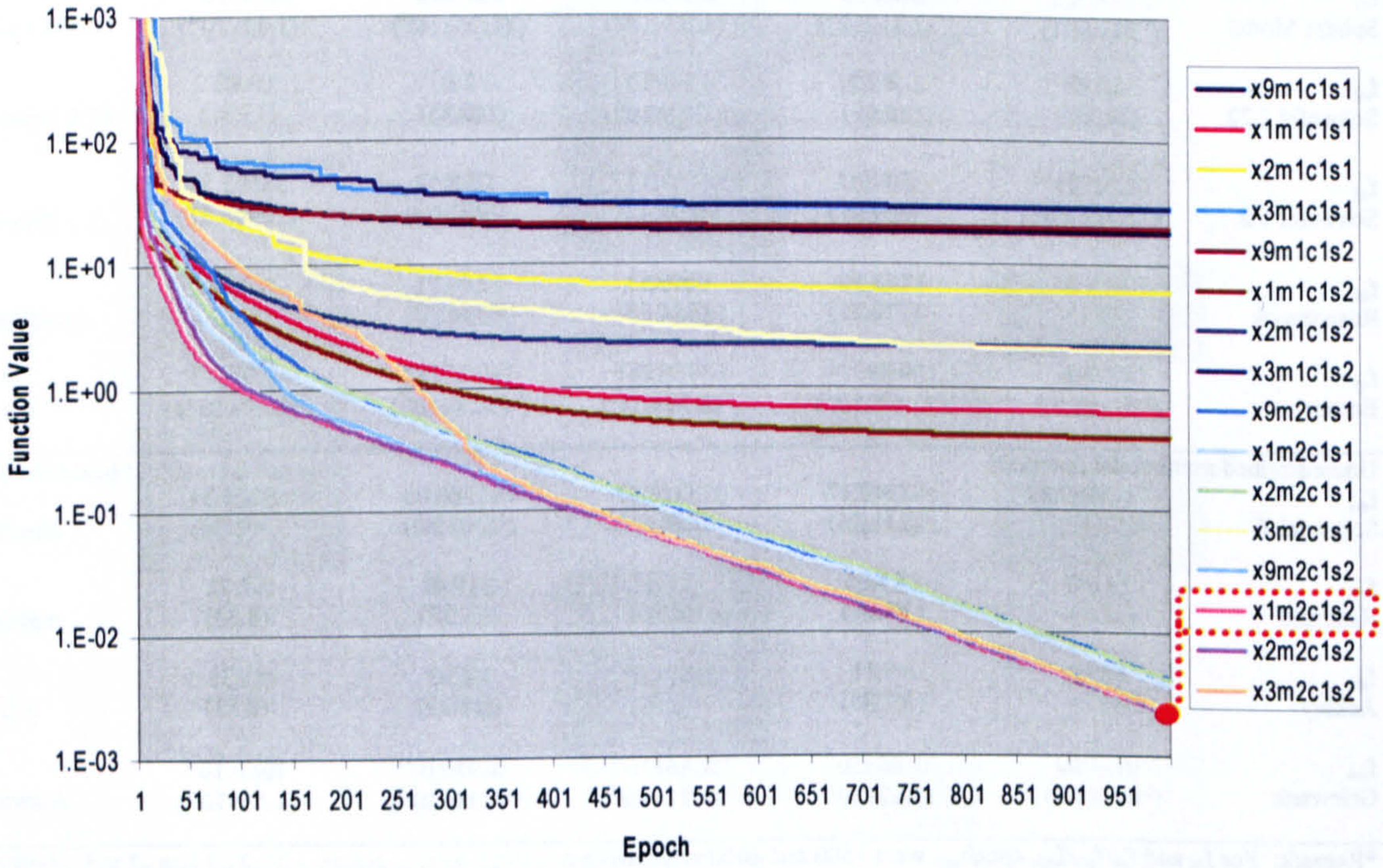


Fig. 7.3. Performance graphs for unconstrained unimodal f_{u2} (solution = 0).

7.2.3 Effectiveness and limitations of geometrical recombination

Particular attention was paid to geometrical recombination (XO=3), since its performance might be extremely good or bad in response to the forms of the test functions. For the test function f_{m1} (Schwefel's Function 7), the best results involved geometrical recombination, whichever options of mutation or selection operators were used, as shown in Tables 7.1 to 7.9 (except Table 7.2). The performance was still the best even in situations of reduced population and/or epoch. For the test function f_{u4} (Generalized Rosenbrock's Function), geometrical recombination also gave the best results in association with the combination of [MU=2 + SE=2]. The test function f_{u4} is shown in Eq (7.3) before, while f_{m1} as follows (refer to Eq (A18) in Appendix I for detail).

$$f_{m1}(x) = \sum_{i=1}^{10}[-x_i \sin(\sqrt{|x_i|})] \qquad (7.4)$$

For instance, as f_{m1} has an asymmetrical component $-x_i \sin(\sqrt{|x_i|})$, the "root-ratio-square" nature of geometrical recombination would continually promote the search no matter whether the variables were positive or negative. Therefore, it would enhance the effectiveness of EA as compared to other kinds of recombination, and a more satisfactory performance could be achieved. The performance graphs of f_{m1} in Figs. 7.4 and 7.5 are used to show the robustness of geometrical recombination (XO=3), at the conditions of full or reduced population and/or epoch. The combination of [XO=3 + MU=2 + CH=1 + SE=2] provided the best performance.

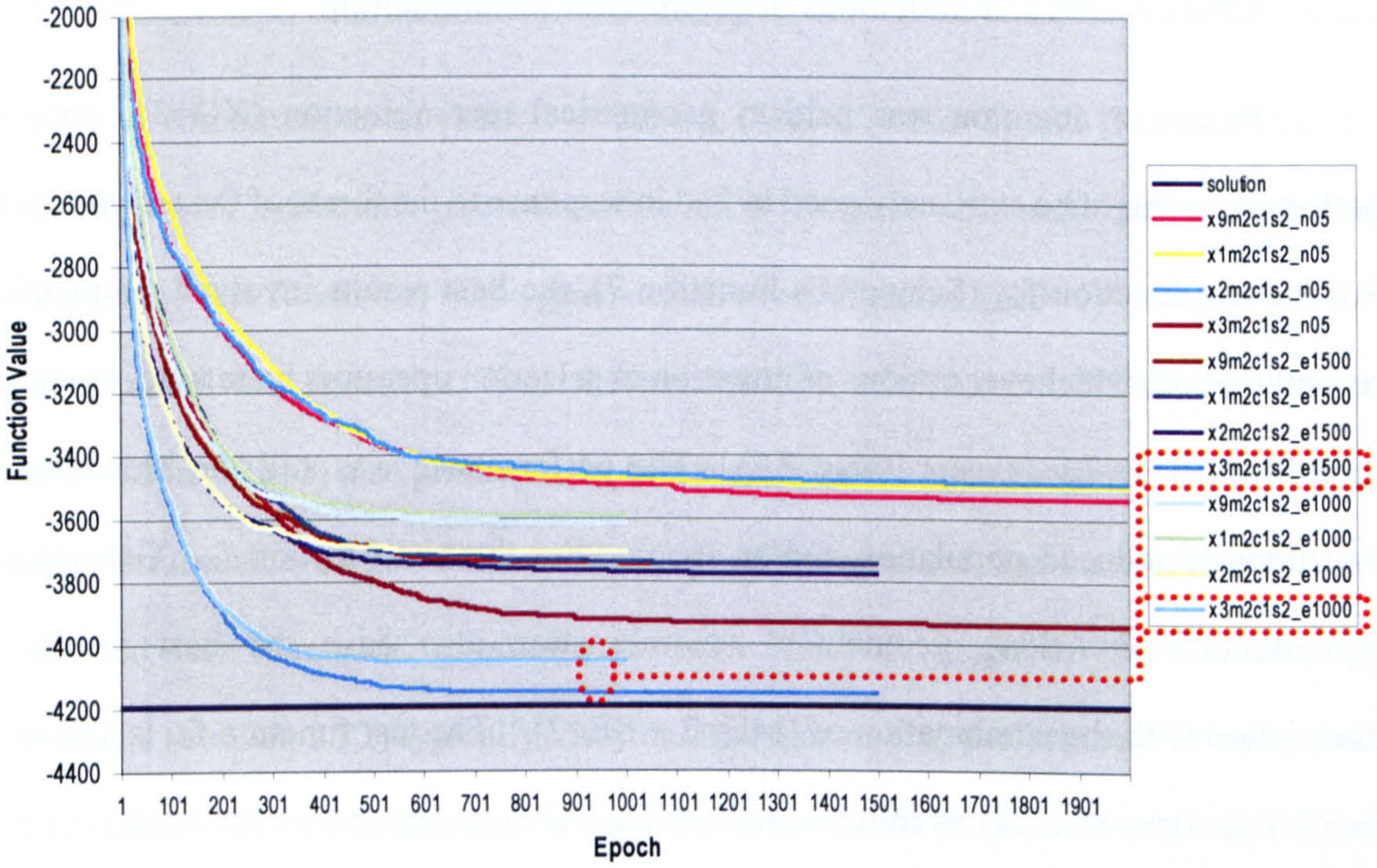


Fig. 7.4. Performance graphs at reduced population/epoch for unconstrained multimodal f_{m1}.

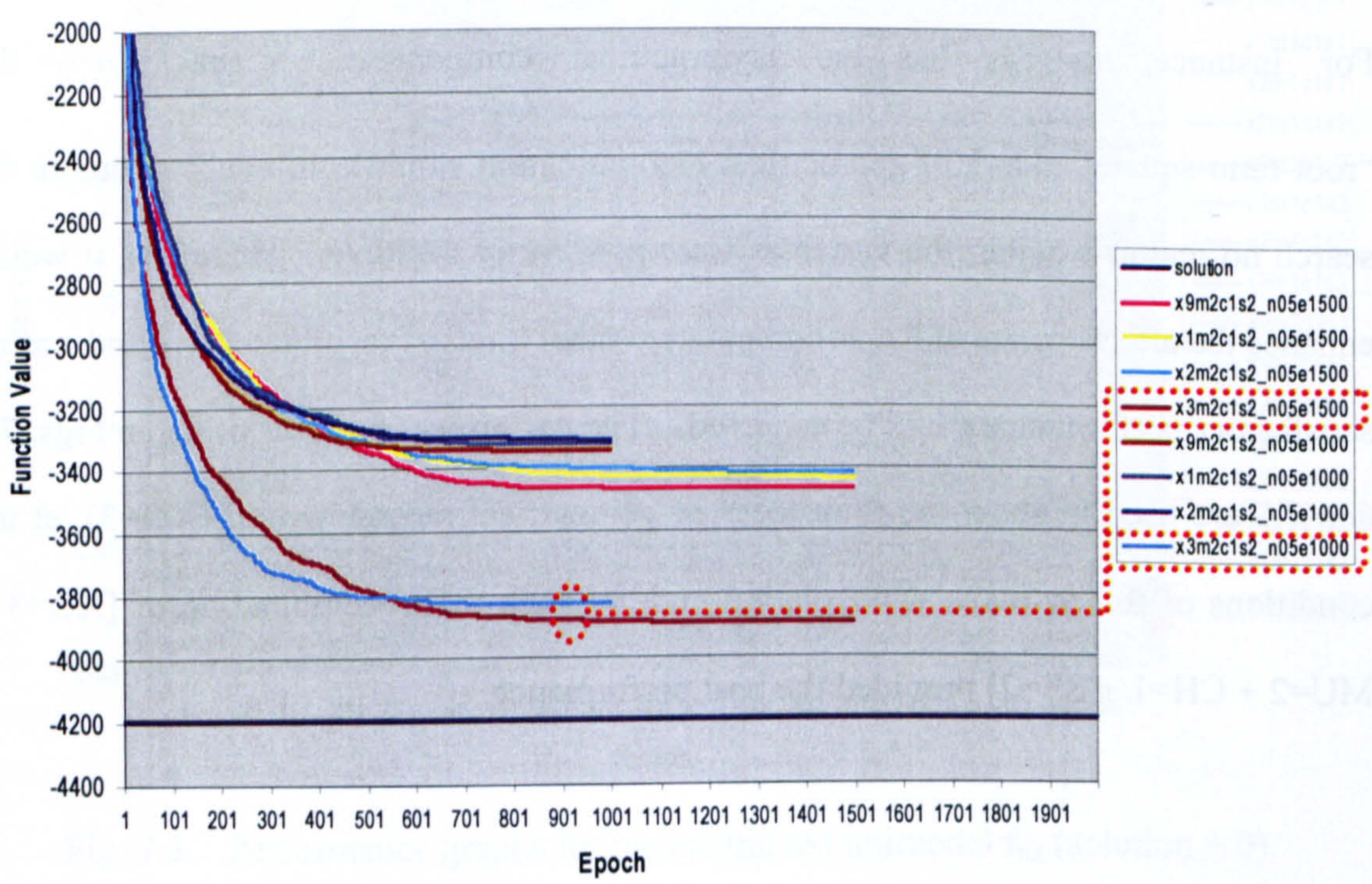


Fig. 7.5. Performance graphs at reduced population and epoch for unconstrained multimodal f_{m1}.

On the contrary for the test function f_{u3} (Schwefel's Problem 1.2), the geometrical recombination had premature convergence far from the optimum, as shown in Tables 7.1 to 7.9. Such extremely bad results of f_{u3} occurred because of geometrical recombination's inherently absolute effect of "root-ratio-square" to the problem variables, since the test function f_{u3} has the component $\sum_{j=1}^{i} x_j$ which would determine the actual summation result of all the variables. If the variables were forced to be positive through "root-ratio-square", the summation result would inevitably become larger. Although mutation might evolve the individual from positive to negative (or vice versa), this would not be helpful. The performance curve of f_{u3} in Fig. 7.6 is used to illustrate this problem, and no matter which combination of mutation and selection was used, the effect of geometrical recombination (XO=3) was dominant and early premature convergence happened in all those cases. In Fig. 7.6, the four graphs related to geometrical recombination overlapped at the premature convergence trajectories, which were at a level which was far from the solution of zero.

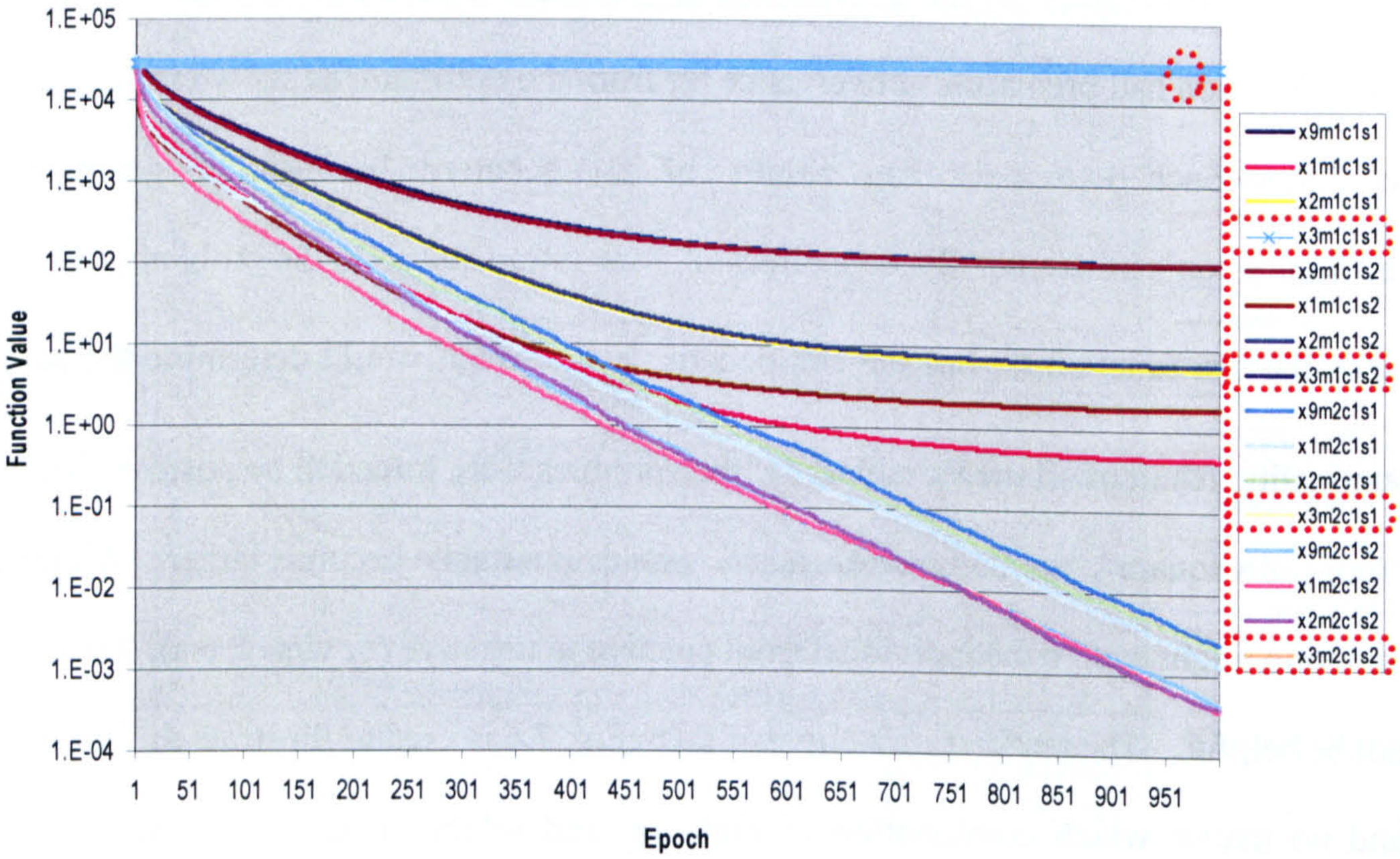


Fig. 7.6. Performance graphs for the unconstrained unimodal f_{u3} (solution = 0).

7.2.4 Conclusion

Due to the extremes in performance of geometrical recombination for unconstrained problems, excellent on the one hand but fatal on the other due to inherent "root-ratio-square" effect, a thorough understanding of the evaluation function and the relationship of the problem variables was crucial. However, the evaluation function might be more difficult to envision if the HVAC optimization problem was developed through the plant simulation model, as then its nature and characteristics could not be so easily assessed. In this regard, geometrical recombination (XO=3) was not recommended, since arithmetic recombination (XO=1) could still provide satisfactory results in general. But geometrical recombination was still held in reserve, as it could be still considered for application if its performance in a trial run was comparable to or better than that of arithmetic recombination.

7.3 Effectiveness at Reduced Population and Epoch

At the initial stage of the experiments, the population and epoch were typically 10 and 1000 respectively. The population of 10 was used rather than the typically larger populations of tens or hundreds for a typical GA, since one of the primary objectives of the experiments was to identify the most robust approach with minimum evaluation function calls. This is critical to determine the efficiency of EA for the practical engineering optimization problems, particularly as the evaluation would be carried out by a plant simulation model. Since the appropriate population size and span of epoch in handling real life engineering optimization problems are generally unknown beforehand, an effective and efficient EA should still give near-optimal or acceptable results at a reduced population and epoch. In Section 7.1, the combination of [MU=2 + SE=2] gave good results with different kinds of test functions, hence further analysis was required to determine the option of recombination operators which would provide the most satisfactory and stable results under the more stringent conditions of the reduced population and/or epoch.

7.3.1 Performance at reduced population or reduced epoch

At the reduced population of 5, but with the original epoch (Table 7.5), arithmetic recombination (XO=1) gave the best performance in general. On the other hand, at the reduced epoch of 750 and 500 (i.e. three quarters and half of the original epoch respectively) but with the original population (Tables 7.6 and 7.7 respectively), the performance of different recombination operators was comparable for the constrained test functions. Arithmetic recombination (XO=1) still assisted in giving the best results for the unconstrained test functions. The graphs of test function f_{m2} (Generalized

Rastrigin's Function) are used to illustrate the performance of arithmetic recombination at these limited conditions, as shown in Fig. 7.7. The combination of [XO=1 + MU=2 + CH=1 + SE=2] provided the best performance in each scenario.

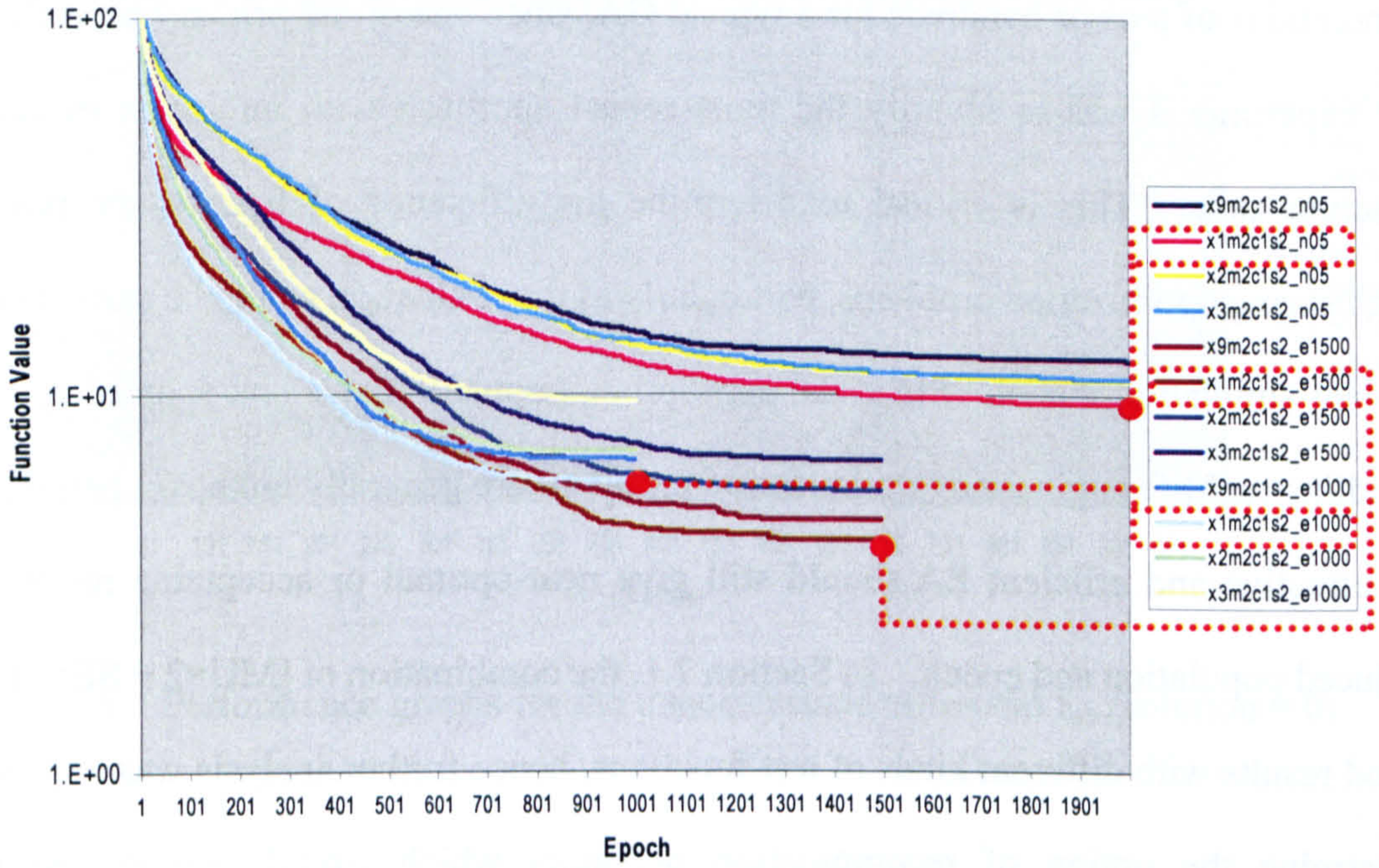


Fig. 7.7. Performance graphs for unconstrained multimodal test function f_{m2}.

7.3.2 Performance at reduced population and reduced epoch (down to three-quarters)

At both the reduced population of 5 and the reduced epoch of 750 (Table 7.8), geometrical recombination (XO=3) gave satisfactory results overall for the constrained test functions, as illustrated by test function f_{c6} (Hock and Schittkowski's Heat Exchanger Design) in Fig. 7.8. The combination of [XO=3 + MU=2 + CH=1 + SE=2] could have the best performance for the reduced epoch of 2250 (three-quarters of the original epoch).

Again arithmetic recombination (XO=1) gave the overall satisfactory results for the unconstrained test functions, as illustrated from test function f_{m3} (Ackley's Function) in Fig. 7.9. The combination of [XO=1 + MU=2 + CH=1 + SE=2] could provide the

best performance at epoch$_{max}$ of 750 and 500.

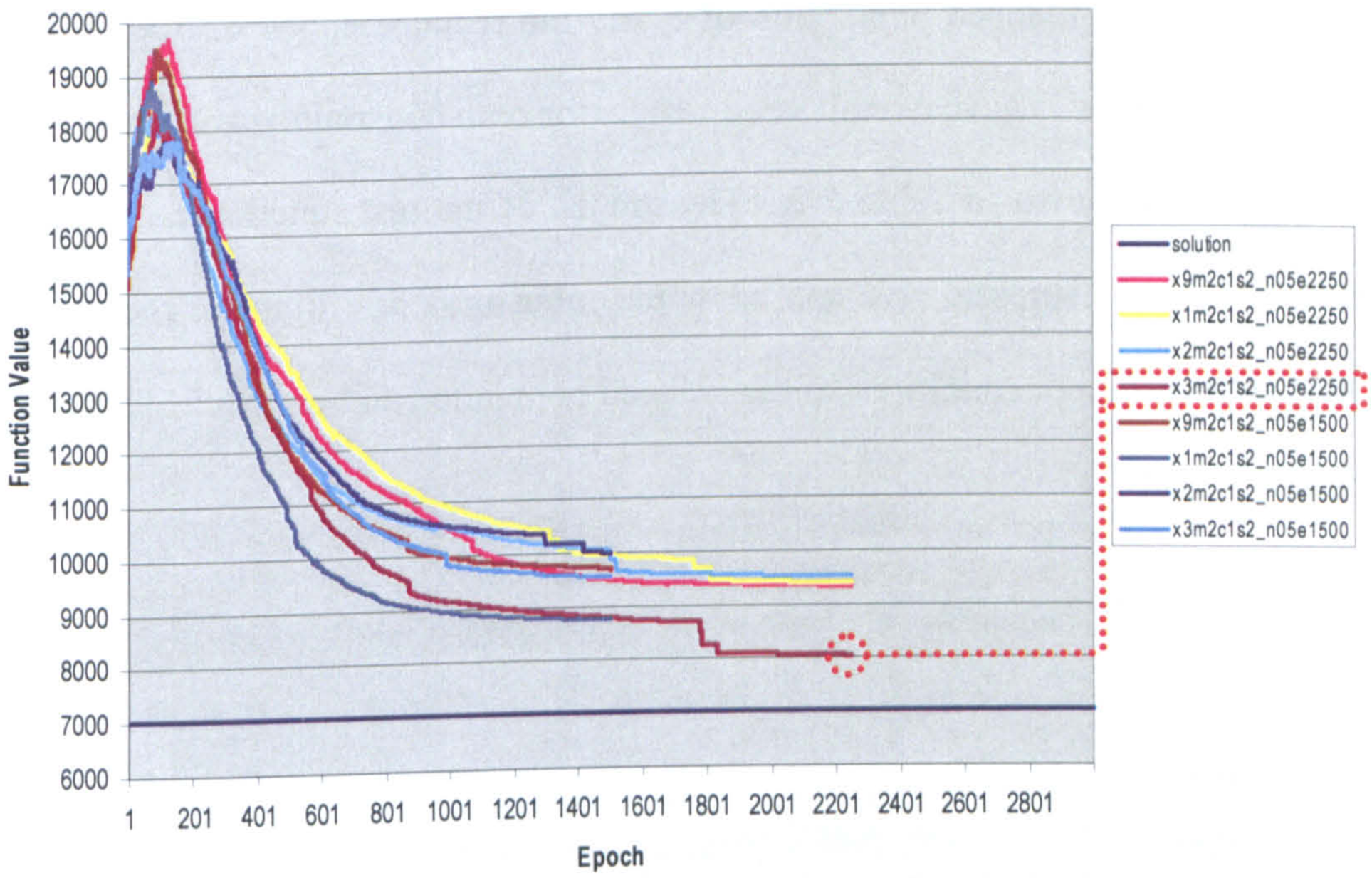


Fig. 7.8. Performance graphs at both reduced population and epoch for constrained f_{c6}.

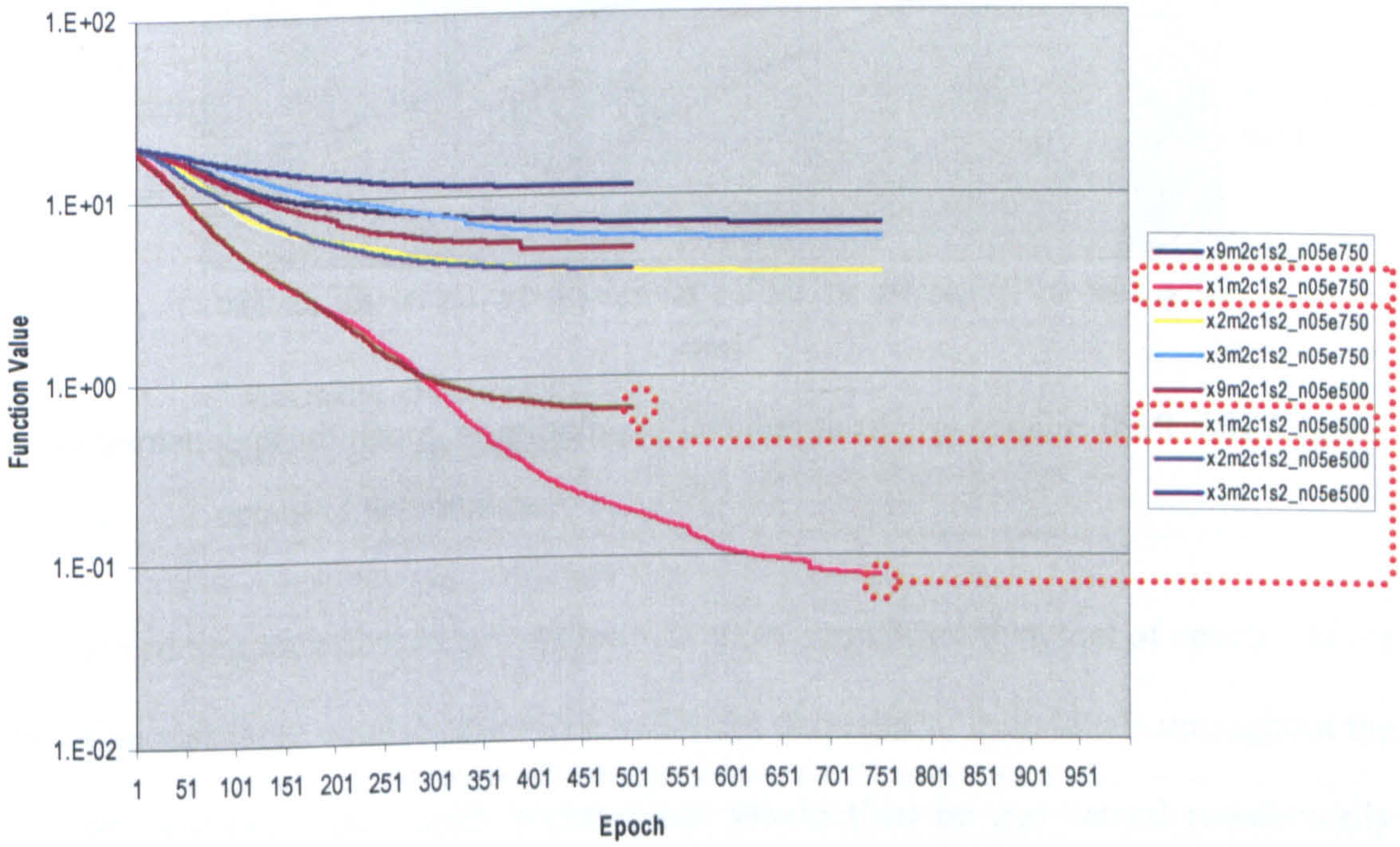


Fig. 7.9. Performance graphs at both reduced population and epoch for unconstrained multimodal f_{m3}.

7.3.3 Performance at reduced population and reduced epoch (down to half)

At both the reduced population of 5 and the reduced epoch of 500, arithmetic recombination (XO=1) gave overall good results for both constrained and unconstrained test functions as shown in Table 7.9. The graphs of the test function f_{u2} (Schwefel's Problem 2.22) are representative and show the robustness of arithmetic recombination (XO=1) at the different conditions of the reduced population and/or epoch. This is shown in Figs. 7.10 and 7.11.

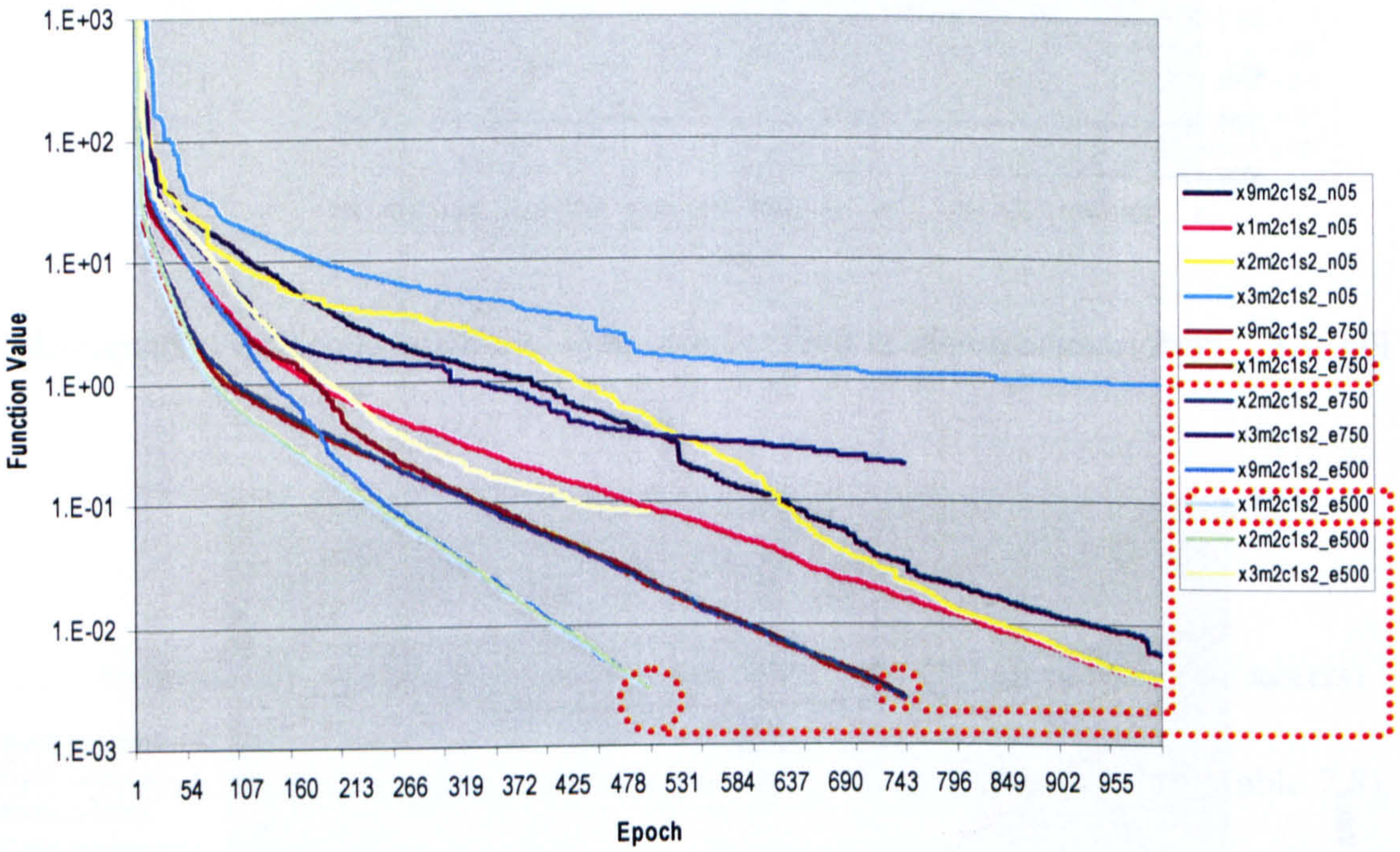


Fig. 7.10. Performance graphs at reduced population or epoch for unconstrained f_{u2}.

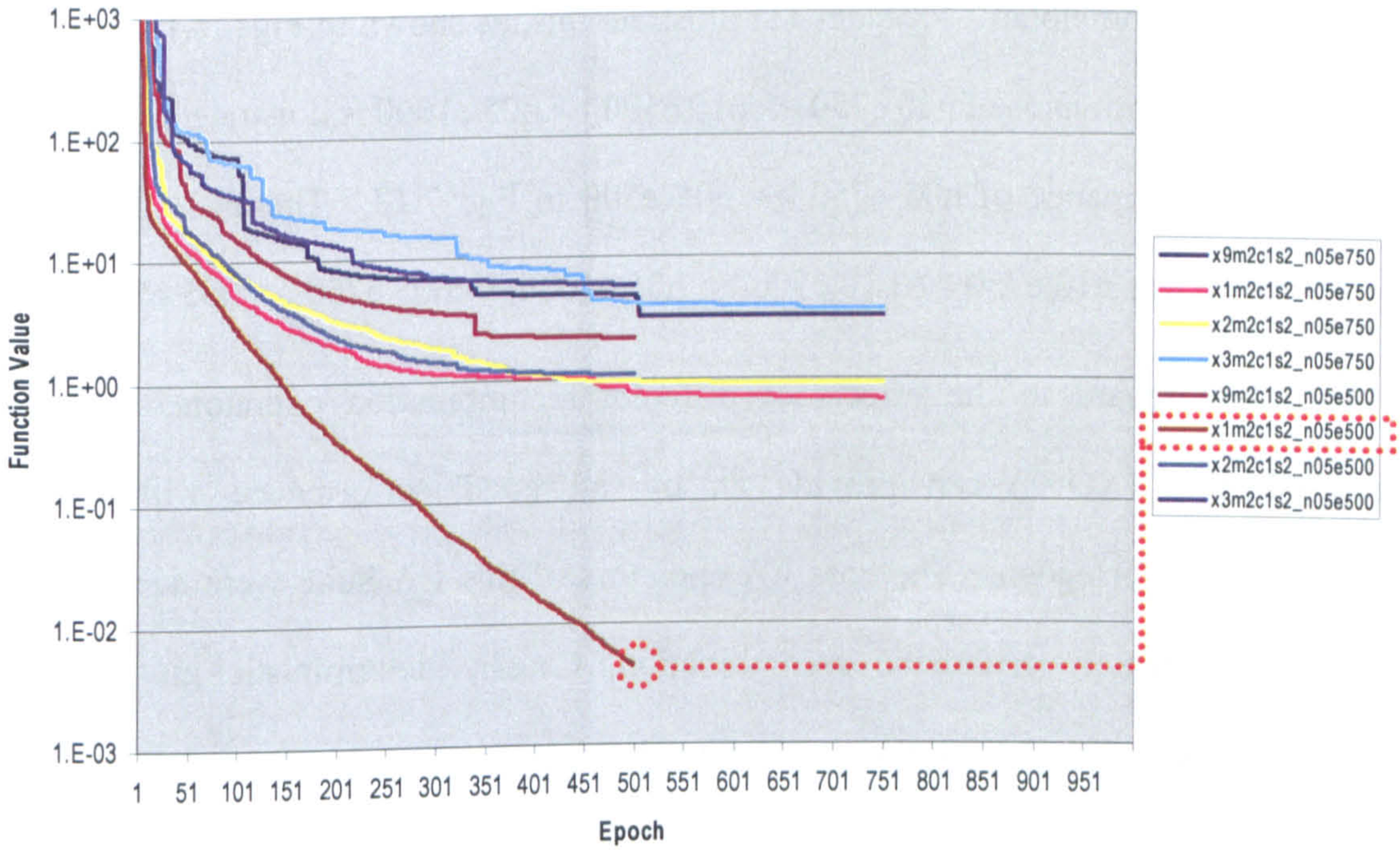


Fig. 7.11. Performance graphs at reduced population and epoch for unconstrained unimodal f_{u2}.

7.3.4 Conclusion

Among the different cases of the reduced population and/or epoch, the proximity to the respective solutions was given typically by the following sequence:

n10 e750 >> n10 e500 >> n05 e1000 >> n05 e750 >> n05 e500

where >>: "was more effective than"
n: number of population
e: epoch of termination

This showed that the effect of population was more significant than that of epoch. If the population was large enough, there is a sufficient diversity of individuals throughout the evolution process. The search performance would then be maintained satisfactorily even at a condition of reduced epoch. On the other hand, a sufficient span of epoch was important to let the search approach the global or near-optimum. The graphs of the test

function f_{c2} (Himmelblau's Problem 11) illustrate this, as shown in Figs. 7.12 and 7.13. The overall performance of n10 e750 >> n10 e500 >> n05 e1000 is illustrated in Fig. 7.12, while the performance of n05 e750 >> n05 e500 in Fig. 7.13. Therefore the overall performance was n10 e750 >> n10 e500 >> n05 e1000 >> n05 e750 >> n05 e500.

With regard to the effects of different recombination operators, arithmetic recombination (XO=1) can provide an overall good performance with reduced population and/or epoch. The core EA operators of this EA Suite were derived from these tests, namely arithmetic recombination, Cauchy deterministic mutation and tournament selection [XO=1 + MU=2 + SE=2].

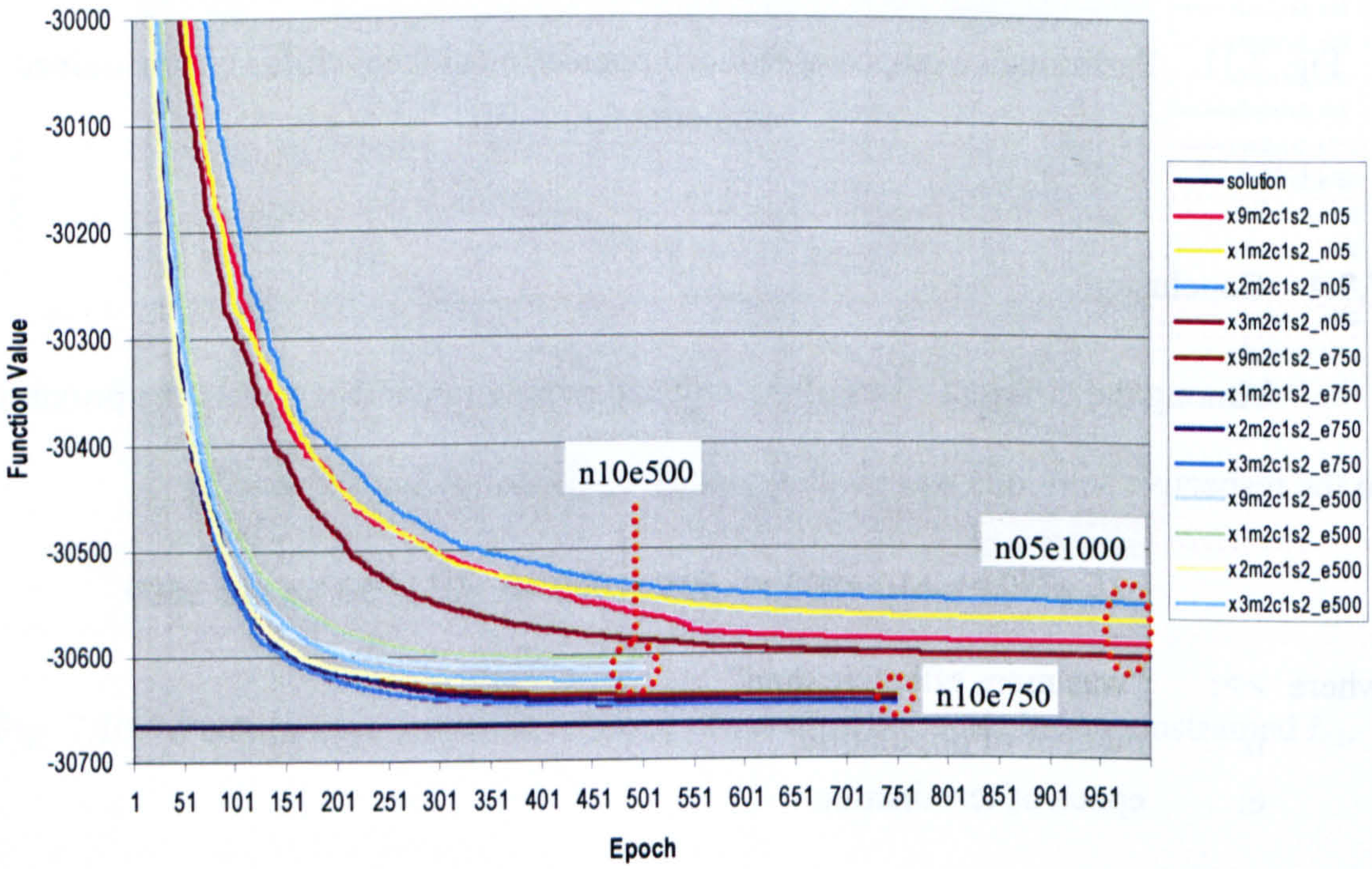


Fig. 7.12. Performance graphs at reduced population or epoch for constrained f_{c2}.

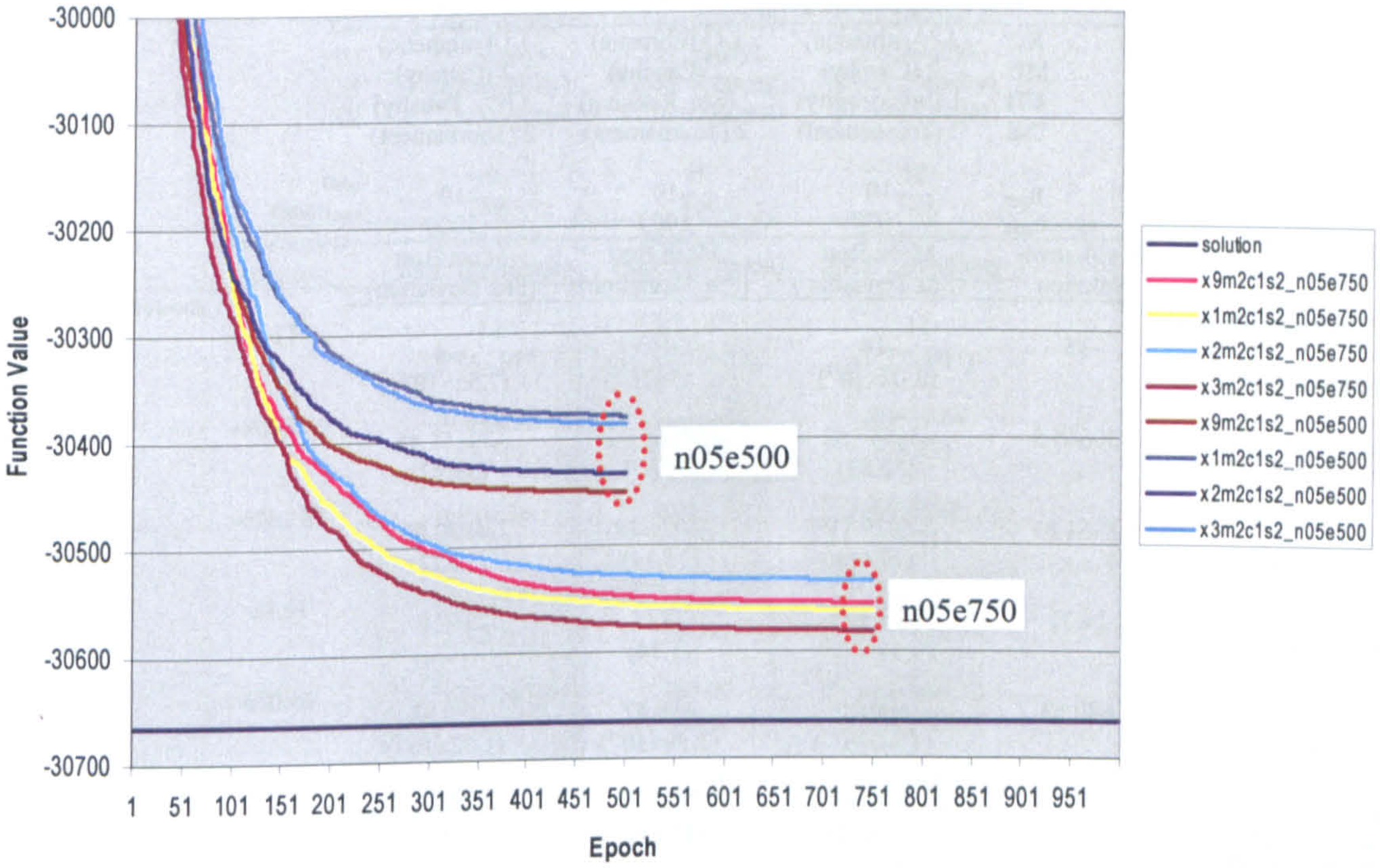


Fig. 7.13. Performance graphs at reduced population and epoch for constrained f_{c2}.

7.4 Performance of Constraint Handling Operators

As discussed in Sections 7.1 to 7.3, the combination of arithmetic recombination, Cauchy deterministic mutation and tournament selection [XO=1 + MU=2 + SE=2] gave overall a good optimization performance for a variety of tests. On the other hand, since the geometrical recombination (XO=3) might still provide satisfactory results for certain types of test functions, the combination with this recombination operator [XO=3 + MU=2 + SE=2] was also used in the study of the effectiveness of different constraint handling operators. As shown in Tables 7.10 to 7.15 [XO=1 + MU=2 + SE=2] and Tables 7.16 to 7.21 [XO=3 + MU=2 + SE=2], all the methods showed generally comparable performance. However from the performance graphs, stochastic ranking (CH=2) and dynamic penalty (CH=3) had slightly worse results at the reduced population and/or epoch for test functions f_{c3}, f_{c4} and f_{c5} (Floundas and Pardalos, Hock and Schittkowski's Problem 113 and Hock and Schittkowski's Problem 100 respectively).

Table 7.10. Performance at typical population and epoch (arithmetic recombination)

	XO	1 (Arithmetic)	1 (Arithmetic)	1 (Arithmetic)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	2 (Sto. Ranking)	3 (Dy. Penalty)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	10	10	10
	$epoch_{max}$	1000	1000	1000
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions				
f_{c1} Floundas & P	-15	-14 (8.02×10^{-1})	-14 (7.49×10^{-1})	-14 (7.56×10^{-1})
f_{c2} Himmelblau 11	-30665.5	-30647.53 (52.83)	-30655.89 (38.87)	-30655.48 (33.42)
f_{c3} Floundas & P	-6961.81	-6929.11 (17.61)	-6936.68 (15.14)	-6930.00 (18.22)
f_{c4} Hock & S 113	24.31	25.89 (9.90×10^{-1})	26.11 (1.14)	26.10 (1.22)
f_{c5} Hock & S 100	680.63	680.85 (1.34×10^{-1})	680.87 (1.59×10^{-1})	680.88 (1.72×10^{-1})
f_{c6} Hock & S HX ($epoch_{max}$ = 3000)	7049.33	7611.68 (638.52)	7627.51 (665.02)	7880.06 (1166.03)
f_{c7} Maa & Shanblatt	0.75	0.7487 (1.84×10^{-3})	0.7488 (2.95×10^{-3})	0.7486 (1.40×10^{-3})

Table 7.11. Performance at half population (arithmetic recombination)

	XO	1 (Arithmetic)	1 (Arithmetic)	1 (Arithmetic)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	2 (Sto. Ranking)	3 (Dy. Penalty)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	5	5	5
	$epoch_{max}$	1000	1000	1000
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions				
f_{c1} Floundas & P	-15	-13 (9.98×10^{-1})	-13 (9.58×10^{-1})	-13 (1.05)
f_{c2} Himmelblau 11	-30665.5	-30567.74 (111.14)	-30567.37 (142.54)	-30588.24 (132.56)
f_{c3} Floundas & P	-6961.81	-6845.92 (72.92)	-6852.62 (64.16)	-6842.45 (63.85)
f_{c4} Hock & S 113	24.31	25.47 (8.84)	28.19 (4.98)	28.98 (7.80)
f_{c5} Hock & S 100	680.63	681.62 (9.20×10^{-1})	668.16 (96.43)	681.77 (1.00)
f_{c6} Hock & S HX ($epoch_{max}$ = 3000)	7049.33	8514.78 (3452.18)	8438.22 (2547.08)	8108.24 (3376.35)
f_{c7} Maa & Shanblatt	0.75	0.8037 (3.72×10^{-2})	-0.7980 (4.14×10^{-2})	-0.7843 (1.20×10^{-1})

Table 7.12. Performance at three-quarters epoch (arithmetic recombination)

	XO	1 (Arithmetic)	1 (Arithmetic)	1 (Arithmetic)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	2 (Sto. Ranking)	3 (Dy. Penalty)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	10	10	10
	$epoch_{max}$	750	750	750
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions				
f_{c1} Floundas & P	-15	-14 (8.86×10^{-1})	-14 (8.63×10^{-1})	-14 (9.37×10^{-1})
f_{c2} Himmelblau 11	-30665.5	-30648.12 (53.71)	-30649.89 (45.10)	-30637.64 (78.27)
f_{c3} Floundas & P	-6961.81	-6919.34 (23.88)	-6922.79 (21.94)	-6922.92 (21.41)
f_{c4} Hock & S 113	24.31	26.47 (1.27)	26.69 (1.79)	26.62 (1.72)
f_{c5} Hock & S 100	680.63	680.94 (1.93×10^{-1})	680.99 (1.54×10^{-1})	680.96 (2.10×10^{-1})
f_{c6} Hock & S HX ($epoch_{max}$ = 2250)	7049.33	7813.88 (794.98)	7822.92 (1450.79)	7946.82 (937.62)
f_{c7} Maa & Shanblatt	0.75	0.7528 (1.18×10^{-2})	0.7517 (9.27×10^{-3})	0.7523 (9.33×10^{-3})

Table 7.13. Performance at half epoch (arithmetic recombination)

	XO	1 (Arithmetic)	1 (Arithmetic)	1 (Arithmetic)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	2 (Sto. Ranking)	3 (Dy. Penalty)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	10	10	10
	$epoch_{max}$	500	500	500
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions				
f_{c1} Floundas & P	-15	-14 (1.09)	-14 (8.84×10^{-1})	-14 (1.03)
f_{c2} Himmelblau 11	-30665.5	-30603.78 (112.82)	-30638.75 (58.87)	-30632.47 (77.36)
f_{c3} Floundas & P	-6961.81	-6904.97 (25.89)	-6894.99 (39.34)	-6898.70 (33.21)
f_{c4} Hock & S 113	24.31	30.51 (14.14)	28.08 (2.91)	28.13 (4.10)
f_{c5} Hock & S 100	680.63	681.13 (32.40)	681.16 (41.30)	681.17 (3.34×10^{-1})
f_{c6} Hock & S HX ($epoch_{max}$ = 1500)	7049.33	8275.74 (2015.17)	8146.04 (1902.40)	8623.38 (2222.97)
f_{c7} Maa & Shanblatt	0.75	0.7718 (2.68×10^{-2})	0.7701 (2.92×10^{-2})	0.7780 (2.89×10^{-2})

Table 7.14. Performance at half population and three-quarters epoch (arithmetic recombination)

	XO	1 (Arithmetic)	1 (Arithmetic)	1 (Arithmetic)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	2 (Sto. Ranking)	3 (Dy. Penalty)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	5	5	5
	$epoch_{max}$	750	750	750
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions				
f_{c1} Floundas & P	-15	-13 (9.29×10^{-1})	-13 (8.57×10^{-1})	-13 (9.95×10^{-1})
f_{c2} Himmelblau 11	-30665.5	-30561.93 (130.38)	-30505.79 (215.03)	-30491.70 (190.62)
f_{c3} Floundas & P	-6961.81	-6771.17 (124.96)	-6782.49 (89.88)	-6795.95 (106.59)
f_{c4} Hock & S 113	24.31	32.10 (10.90)	28.82 (8.71)	35.68 (18.47)
f_{c5} Hock & S 100	680.63	668.63 (96.49)	668.67 (96.50)	682.37 (1.30)
f_{c6} Hock & S HX ($epoch_{max}$ = 2250)	7049.33	9386.44 (4271.41)	9415.87 (3380.91)	8788.00 (3171.95)
f_{c7} Maa & Shanblatt	0.75	0.8047 (1.26×10^{-1})	0.8313 (4.56×10^{-2})	0.7973 (1.71×10^{-1}0

Table 7.15. Performance at half population and half epoch (arithmetic recombination)

	XO	1 (Arithmetic)	1 (Arithmetic)	1 (Arithmetic)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	2 (Sto. Ranking)	3 (Dy. Penalty)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	5	5	5
	$epoch_{max}$	500	500	500
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions				
f_{c1} Floundas & P	-15	-12 (1.04)	-12 (1.17)	-12 (1.12)
f_{c2} Himmelblau 11	-30665.5	-30378.70 (214.04)	-30375.05 (313.54)	-30438.44 (241.06)
f_{c3} Floundas & P	-6961.81	-6560.40 (376.55)	-6506.91 (801.12)	-6508.67 (424.38)
f_{c4} Hock & S 113	24.31	92.53 (141.68)	93.13 (140.59)	98.31 (148.29)
f_{c5} Hock & S 100	680.63	684.72 (4.74)	657.45 (135.61)	683.86 (1.93)
f_{c6} Hock & S HX ($epoch_{max}$ = 1500)	7049.33	8770.05 (3991.54)	9205.15 (3646.09)	9583.12 (4542.87)
f_{c7} Maa & Shanblatt	0.75	0.8699 (5.42×10^{-2})	0.8430 (1.37×10^{-1})	0.8785 (5.16×10^{-2})

Table 7.16. Performance at typical population and epoch (geometrical recombination)

	XO	3 (Geometrical)	3 (Geometrical)	3 (Geometrical)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	2 (Sto. Ranking)	3 (Dy. Penalty)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	10	10	10
	$epoch_{max}$	1000	1000	1000
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions				
f_{c1} Floundas & P	-15	-14 (8.03×10^{-1})	-14 (8.09×10^{-1})	-14 (8.13×10^{-1})
f_{c2} Himmelblau 11	-30665.5	-30640.65 (63.43)	-30662.62 (18.77)	-30647.42 (62.16)
f_{c3} Floundas & P	-6961.81	-6927.57 (17.11)	-6939.13 (16.08)	-6931.77 (17.02)
f_{c4} Hock & S 113	24.31	25.89 (9.54×10^{-1})	26.18 (1.13)	26.41 (1.13)
f_{c5} Hock & S 100	680.63	685.79 (2.41)	685.98 (2.36)	686.10 (2.52)
f_{c6} Hock & S HX ($epoch_{max}$ = 3000)	7049.33	8079.18 (1361.11)	7862.54 (1242.34)	7774.09 (734.87)
f_{c7} Maa & Shanblatt	0.75	0.7492 (2.91×10^{-3})	0.7483 (9.42×10^{-4})	0.7485 (1.85×10^{-3})

Table 7.17. Performance at half population (geometrical recombination)

	XO	3 (Geometrical)	3 (Geometrical)	3 (Geometrical)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	2 (Sto. Ranking)	3 (Dy. Penalty)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	5	5	5
	$epoch_{max}$	1000	1000	1000
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions				
f_{c1} Floundas & P	-15	-13 (1.14)	-13 (8.92×10^{-1})	-13 (1.05)
f_{c2} Himmelblau 11	-30665.5	-30600.94 (103.87)	-30543.27 (167.91)	-30568.66 (156.60)
f_{c3} Floundas & P	-6961.81	-6846.05 (76.34)	-6848.70 (58.57)	-6837.40 (64.66)
f_{c4} Hock & S 113	24.31	27.07 (7.71)	27.99 (3.76)	27.81 (6.66)
f_{c5} Hock & S 100	680.63	672.96 (97.15)	673.66 (97.27)	674.06 (97.36)
f_{c6} Hock & S HX ($epoch_{max}$ = 3000)	7049.33	8818.18 (2180.84)	8007.76 (2385.35)	7854.54 (3840.59)
f_{c7} Maa & Shanblatt	0.75	0.7992 (3.90×10^{-2})	0.7916 (3.75×10^{-2})	0.8062 (3.84×10^{-2})

Table 7.18. Performance at three-quarters population (geometrical recombination)

	XO	3 (Geometrical)	3 (Geometrical)	3 (Geometrical)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	2 (Sto. Ranking)	3 (Dy. Penalty)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	10	10	10
	$epoch_{max}$	750	750	750
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions				
f_{c1} Floundas & P	-15	-14 (7.67×10^{-1})	-14 (9.47×10^{-1})	-14 (8.96×10^{-1})
f_{c2} Himmelblau 11	-30665.5	-30648.30 (51.63)	-30645.90 (57.06)	-30642.30 (55.63)
f_{c3} Floundas & P	-6961.81	-6921.74 (20.37)	-6922.49 (25.20)	-6921.03 (23.96)
f_{c4} Hock & S 113	24.31	26.86 (1.87)	26.94 (2.01)	26.56 (1.58)
f_{c5} Hock & S 100	680.63	686.85 (2.65)	686.40 (2.49)	686.60 (2.61)
f_{c6} Hock & S HX ($epoch_{max}$ = 2250)	7049.33	7738.23 (1057.43)	8041.86 (1235.81)	7988.98 (1266.27)
f_{c7} Maa & Shanblatt	0.75	0.7520 (1.18×10^{-2})	0.7518 (8.51×10^{-3})	0.7530 (1.03×10^{-2})

Table 7.19. Performance at half epoch (geometrical recombination)

	XO	3 (Geometrical)	3 (Geometrical)	3 (Geometrical)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	2 (Sto. Ranking)	3 (Dy. Penalty)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	10	10	10
	$epoch_{max}$	500	500	500
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions				
f_{c1} Floundas & P	-15	-14 (8.04×10^{-1})	-14 (1.09)	-13 (1.06)
f_{c2} Himmelblau 11	-30665.5	-30624.02 (90.42)	-30627.42 (78.22)	-30608.37 (103.84)
f_{c3} Floundas & P	-6961.81	-6906.90 (26.56)	-6904.71 (33.50)	-6894.20 (39.50)
f_{c4} Hock & S 113	24.31	28.41 (2.70)	28.53 (2.91)	27.14 (1.86)
f_{c5} Hock & S 100	680.63	686.23 (2.27)	686.53 (2.86)	686.98 (3.13)
f_{c6} Hock & S HX ($epoch_{max}$ = 1500)	7049.33	7983.55 (1322.23)	7962.28 (1188.74)	7956.00 (980.28)
f_{c7} Maa & Shanblatt	0.75	0.7713 (2.66×10^{-2})	0.7711 (2.67×10^{-2})	0.7694 (2.51×10^{-2})

Table 7.20. Performance at half population and three-quarters epoch (geometrical recombination)

	XO	3 (Geometrical)	3 (Geometrical)	3 (Geometrical)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	2 (Sto. Ranking)	3 (Dy. Penalty)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	5	5	5
	$epoch_{max}$	750	750	750
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions				
f_{c1} Floundas & P	-15	-13 (1.21)	-12 (1.20)	-13 (1.15)
f_{c2} Himmelblau 11	-30665.5	-30580.51 (154.66)	-30535.16 (152.34)	-30565.34 (131.48)
f_{c3} Floundas & P	-6961.81	-6799.61 (110.47)	-6765.16 (123.53)	-6651.77 (968.11)
f_{c4} Hock & S 113	24.31	28.25 (6.72)	29.59 (6.05)	27.93 (9.40)
f_{c5} Hock & S 100	680.63	674.94 (97.52)	688.42 (5.13)	688.52 (4.02)
f_{c6} Hock & S HX ($epoch_{max}$ = 2250)	7049.33	8057.86 (4628.77)	8557.93 (3580.83)	8473.35 (1942.03)
f_{c7} Maa & Shanblatt	0.75	0.8027 (1.24×10^{-1})	0.8053 (1.72×10^{-1})	0.8369 (5.11×10^{-2})

Table 7.21. Performance at half population and half epoch (geometrical recombination)

	XO	3 (Geometrical)	3 (Geometrical)	3 (Geometrical)
	MU	2 (Cauchy)	2 (Cauchy)	2 (Cauchy)
	CH	1 (Infeasibility)	2 (Sto. Ranking)	3 (Dy. Penalty)
	SE	2 (Tournament)	2 (Tournament)	2 (Tournament)
	n_{pop}	5	5	5
	$epoch_{max}$	500	500	500
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions				
f_{c1} Floundas & P	-15	-12 (1.14)	-12 (1.34)	-12 (1.30)
f_{c2} Himmelblau 11	-30665.5	-30386.06 (273.57)	-30402.88 (283.44)	-30420.79 (279.04)
f_{c3} Floundas & P	-6961.81	-6472.44 (792.71)	-6537.19 (357.12)	-6467.04 (805.76)
f_{c4} Hock & S 113	24.31	31.14 (12.82)	36.05 (14.20)	38.10 (13.59)
f_{c5} Hock & S 100	680.63	686.19 (7.74)	648.17 (165.47)	677.72 (98.37)
f_{c6} Hock & S HX ($epoch_{max}$ = 1500)	7049.33	9549.48 (3857.36)	9345.42 (3778.38)	8545.85 (4730.58)
f_{c7} Maa & Shanblatt	0.75	0.8761 (4.95×10^{-2})	0.8824 (4.96×10^{-2})	0.8632 (6.54×10^{-2})

For the performance of different constraint handling operators in [XO=1 + MU=2 + SE=2] and [XO=3 + MU=2 + SE=2], the performance graphs of test function f_{c5} are used to illustrate this observation, as shown in Fig. 7.14. In Fig. 7.14, the cases with dynamic penalty (CH=3) deviated most from the solutions.

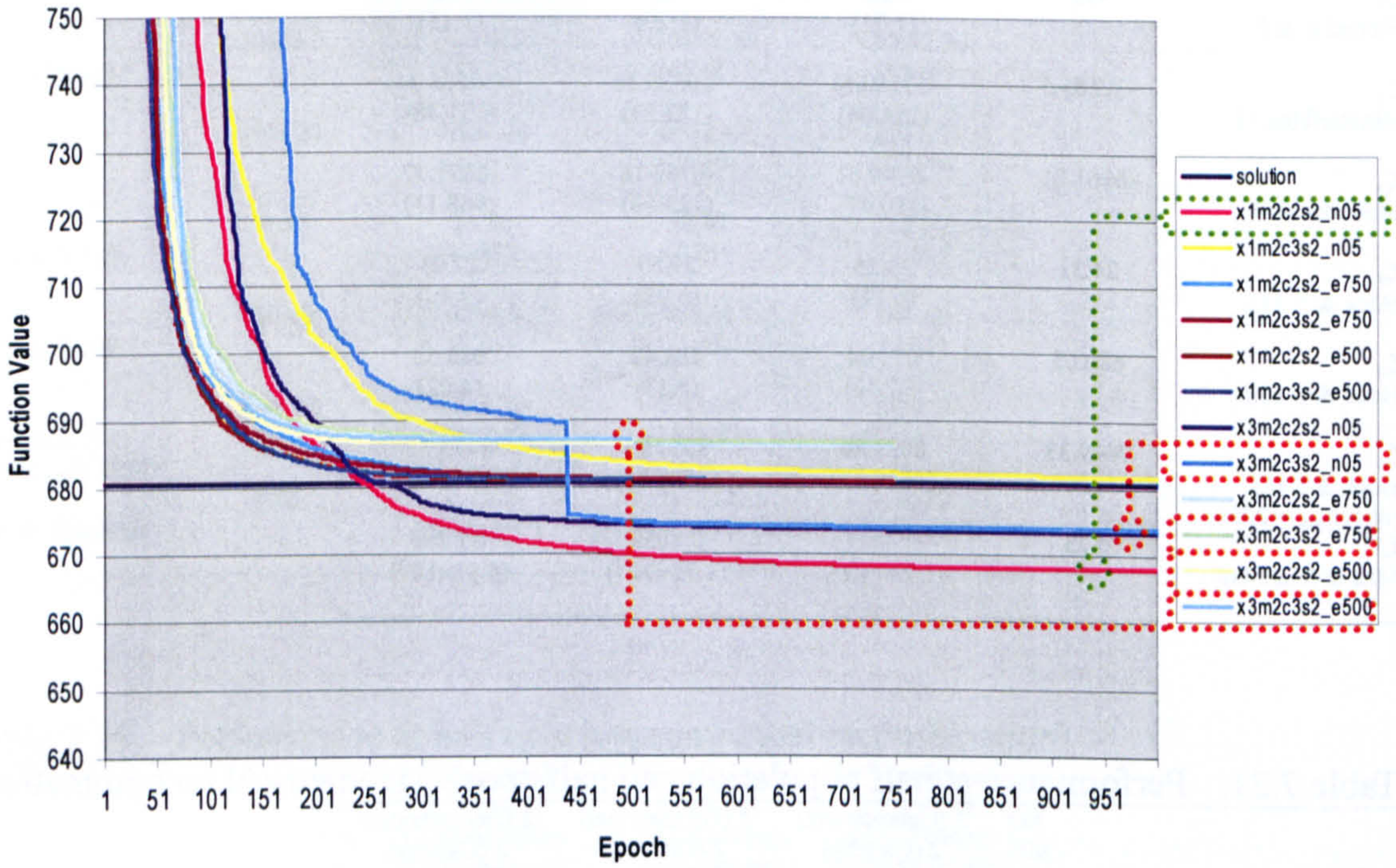


Fig. 7.14. Performance graphs at reduced population or epoch for constrained f_{c5}.

The available options of constraint handling operators provided satisfactory results in general. This was because the "rank-based' approach was applied in the development of the different constraint handling operators in order to prevent any dominating effect of individuals with extreme fitness or constraint values. Based on the relative performances, the choice of constraint handling operator would thus be infeasibility discrimination (CH=1) first, stochastic ranking (CH=2) second, and dynamic penalty (CH=3) last. In the context of this EA Suite, the choice of constraint handling operator was not as important as that of the other EA operators.

7.5 Summary and Conclusion

7.5.1 Summary of findings

Based on the results in the full runs, the series of discussions that were made in Sections 7.1 to 7.4, are summarized as follows.

a. The combination of Cauchy deterministic mutation and tournament selection [MU=2 + SE=2] gave an overall a good result among the different combinations of mutation and selection operators tried. The general performance of Cauchy deterministic mutation (MU=2) was better than Gaussian deterministic mutation (MU=1), while tournament selection (SE=2) performed better than ranking selection (SE=1). In the EA Suite, the effect of mutation operator was more significant than that of the selection operator.

b. For the variety of test functions handled by the EA with Cauchy deterministic mutation, tournament selection and arithmetic recombination [XO=1 + MU=2 + SE=2] gave overall a good result among the recombination operators, even in the more stringent situations of reduced population and/or epoch. On the other hand, geometrical recombination [XO=3 + MU=2 + SE=2] had extremes of performance either very good or very bad, depending on the forms of test functions. This was due to the inherent "absolute" nature of "root-ratio-square" of the geometrical recombination. As a result, the decision to invoke geometrical recombination should be very carefully made, and prior knowledge about the problem function nature is crucial. Therefore in real applications, trial runs for arithmetic and geometrical recombination are required initially, in order to confirm which one produces better optimization results. This was further explored in the example HVAC problems in Chapter 9.

c. The effectiveness and efficiency of an EA search can be enhanced with the inclusion of recombination operators. Therefore in the context of evolutionary programming and evolution strategy, evolution strategy would provide a better overall performance for these kinds of optimization problems. Fong *et al.* (2005b) also illustrated this through a real-world HVAC optimization problem in energy management.

d. The performance of different constraint handling operators was found to be comparable, no matter which combination of [XO=1 + MU=2 + SE=2] or [XO=3 + MU=2 + SE=2] was used. However, infeasibility discrimination (CH=1) provided a relatively stable performance in the variety of constrained test functions used even at the reduced n_{pop} and/or $epoch_{max}$.

7.5.2 Derivation of Robust EA

From these series of experiments, the contribution and effects of different EA operators in searching for the global or near-optimum was assessed. As a whole, the sequence of effectiveness in the contribution of EA optimization is summarized as follows:

Mutation >> Selection >> Recombination
>> Constraint handling (only for constrained problems)

where ">>": "was more effective than".

Consequently, a proposed robust EA (REA) has been derived. The REA with the combination of arithmetic recombination, Cauchy deterministic mutation and tournament selection [XO=1 + MU=2 + SE=2] is recommended for handling the optimization problems with a multimodal, multidimensional, nonlinear and mixed discrete-continuous

nature, typical to the HVAC problems formulated by the plant simulation model. On the other hand, the combination of geometrical recombination, Cauchy deterministic mutation and tournament selection [XO=3 + MU=2 + SE=2] can also be considered if the form of objective function is favourable. The choice of combination can be informed by the outcome of trial runs on the problem.

For constrained problems, infeasibility discrimination (CH=1) is generally recommended as the constraint handling operator, however the choice may not be critical since the performances of different constraint handling operators in the EA Suite were generally comparable. Further study of the effectiveness of these constraint handling operators was carried out; this will be discussed in the forthcoming Section 8.3.

Through the proposed REA, in order to minimize the evaluation function calls of the optimization problem but still obtain satisfactory results, it is recommended that n_{pop} is set at ten. The epoch of termination can be worked out using trial runs for each application to be studied. From the previous experience of Fong *et al.* (2005a, 2005b, 2006) and Hanby *et al.* (2005), $epoch_{max}$ of fifty can be a recommendation in order to have sufficient convergence and to give reliable results for the optimization problems developed by plant simulation models.

7.5.3 Synergetic combination of REA

With the optimal articulation of the EA operators of arithmetic recombination, Cauchy deterministic mutation and tournament selection in REA, good performance originates from the synergetic effect of the EA operators. Exploitation and exploration are continuously maintained in the course of population evolution by REA.

Starting from arithmetic recombination, the parent population produces a

recombined population in which each individual is within the space as bounded by its two parent individuals. This would promote exploitation of better offspring, particularly from two feasible parents.

The recombined population then undergoes the mutation process of deterministic strategy parameter with Cauchy realization. This provides each individual an exploration opportunity in all directions of the multi-dimensional search domain. The search step length is designated from the Cauchy random number and an exponentially decaying deterministic strategy parameter. Inherently this step length of Cauchy realization would be larger than that of Gaussian realization, therefore the continual exploration is possible even when a local optimum is found. In addition, the repair scheme was implemented if any of the variables is mutated out of its respective bounds. Then the restored variables and the in-bound variables would form a new individual in the search space.

The mutated and repaired individuals become the offspring population, and undergo evaluation to acquire the corresponding objective function values (and constraint values for constrained problems), then tournament selection is implemented. At this stage, the REA adopts a plus strategy ($\mu+\mu$) of evolution. Both the μ parents and μ offspring join together to form a tournament population. Each individual of the united population is compared with a number of randomly selected opponents from the joint population. Under this tournament selection, both the objective function value and constraint violation are considered during competition. Therefore the individuals with truly high fitness would have greater probability to be selected, while those with low fitness would not be fully eliminated. The formation of the new population inhibits premature convergence in the search, and infeasible individuals with low fitness may still

exist so the diversity of the search can be maintained.

Elitism is built in the evolution process. After generating the new population for next epoch, the elite individual is chosen by the lowest objective function value for the existence of feasible individuals, or by the lowest constraint violation if the entire population is infeasible. For evolution in next epoch, the elite individual bypasses the operators of recombination and mutation, so that its participation in the subsequent epoch is maintained.

As a whole, owing to the synergetic combination of these EA operators, even the default population size is ten, the REA can still reach the global or near optimum satisfactorily. Unlike the typical GA, it would rely on a large population in tens or hundreds for optimal search, and considerable function calls of evaluation are required for plant simulation problems.

CHAPTER 8 EFFECTIVENESS AND EFFICIENCY OF REA

In this chapter, the in-depth qualitative analysis of the effectiveness and efficiency of the REA is implemented. Since the REA is designed to handle HVAC optimization problems with lengthy computational time, it is necessary to study the reliability and search trend of the REA at limited evaluation function calls. To confirm the effectiveness of REA, benchmarking was conducted with an established EA developed to work with a small population – micro-GA. To understand the contribution of EA operators, particularly the constraint handling operator implemented in the REA, comparative studies were also carried out with the published effective constraint handling techniques.

8.1 Reliability in Reduced Evaluation Function Calls

The operator combination of the proposed REA was also evaluated with respect to its reliability in performing with limited evaluation function calls. With a constant population size (n_{pop} = 10), the evaluation function calls were nearly proportional to the number of epochs of evolution. To confirm the contribution of the REA in this aspect, the changing profile of the objective function value with epoch should have a converging trend towards the global optimum. In this regard, the changing profiles of the constrained test functions involved in the experimentation are plotted together on the same graph as shown in Fig. 8.1. In order to show their performance in a collective way, the normalized objective function values with respect to the corresponding solutions were used. In Fig. 8.1, it can be seen that in less than half of the stipulated epoch of termination ($epoch_{max}$ = 1000 for these functions), even the test function f_{c4} (yellow line) was within the zone of ±10% of the global optimum, as indicated by the semi-transparent

red band in Fig. 8.1.

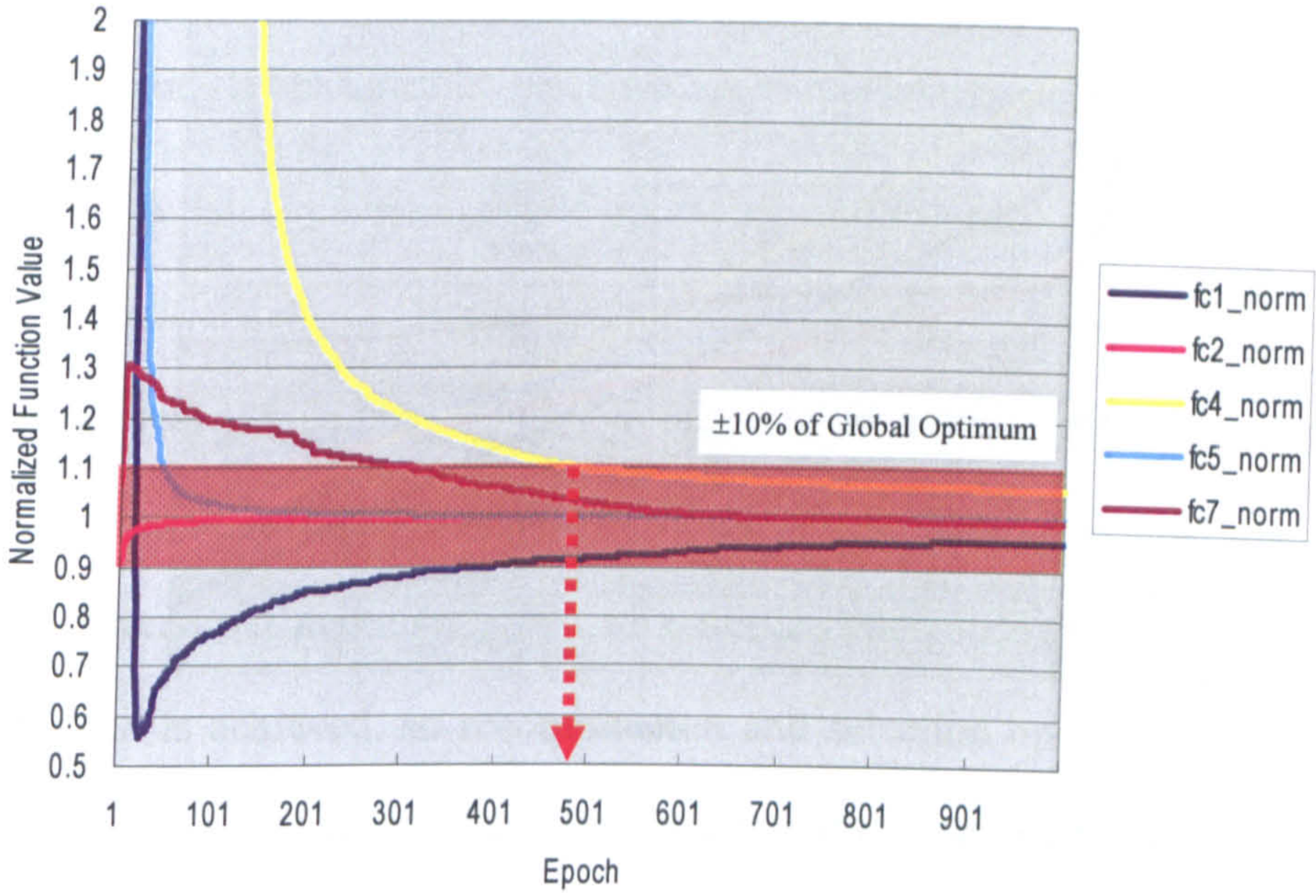


Fig. 8.1. Convergence profile of constrained test functions by using REA

Similar performance of the REA was also found in the unconstrained test functions tested. The unconstrained test functions with the global optimum of zero and $epoch_{max}$ of 1000 are plotted on the same graph as shown in Fig. 8.2. Again in less than half of $epoch_{max}$, the function values were below one, and the search continued, with exponentially decreasing function values along the epoch.

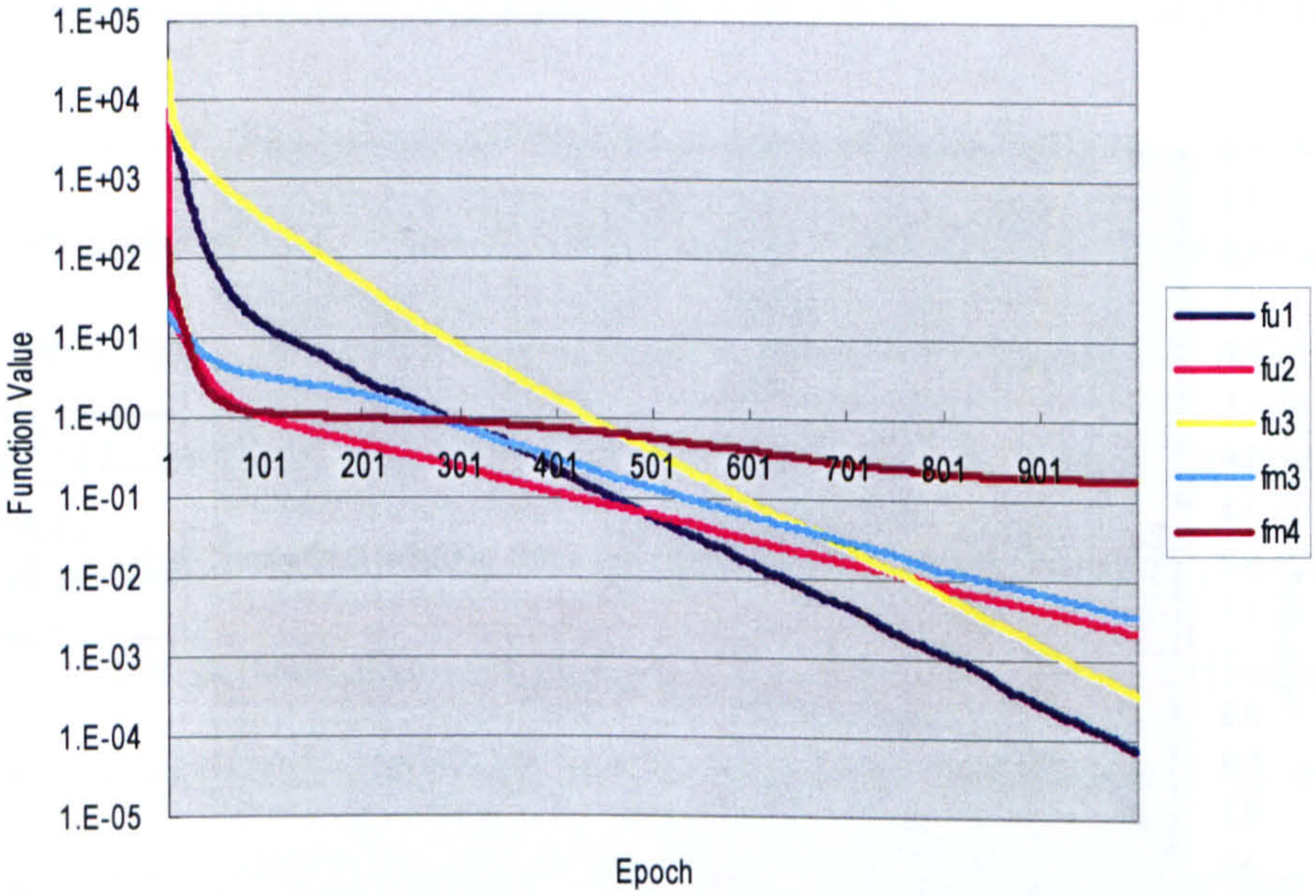


Fig. 8.2. Convergence profile of unconstrained test functions by using REA

8.2 Benchmarking with Micro-GA

8.2.1 Micro-GA in context

The micro-GA (MGA) has been applied to a number of optimization problems. Krishnakumar (1989) used MGA for optimization of flight path angle, Johnson and Abushagur (1995) for optical dielectric gratings, Xiao and Hatsuo (1998) for imaging optimization, and Senecal *et al.* (2000) for optimal design of internal combustion engine. The common reason to adopt MGA as an optimization tool was due to its inherent capability of global search originated from GA, but its very small population is thus less demanding of evaluation function calls. Therefore the MGA is suitable for the optimization problems with considerable running time and is particularly useful for real time applications. For instance, Senecal *et al.* (2000) developed a 3D simulation model of direct injection diesel engine by using a CFD (computational fluid dynamics) code. Function evaluation became crucial in their work since the computational cost of this

process was high. The MGA may also be applicable to the HVAC optimization problems. As a result, benchmarking of the proposed REA with MGA was carried out in order to understand its optimization performance more clearly.

The MGA has three core genetic operators; uniform recombination, tournament selection and reinitialization. The population size is usually five, which is very small as compared to typical simple GA. Owing to this tiny population, elitism is adopted and the best individual in each epoch retained. In the MGA, there is no mutation operator, since enough diversity is maintained through reinitialization once the convergence of the micro-population is achieved, so recombination and selection operators are sufficient. For a population of five, reinitialization gives four new individuals, rather than five, since the elite individual is carried forward to the next epoch. To implement MGA, it is necessary to define the condition of convergence of the micro-population. Senecal (2000) explained that convergence was measured by the progression towards chromosome uniformity in the MGA, and he suggested that convergence occurs when 95% of the genes in the population had the same value. In the implementation of this MGA, the idea of convergence from Senecal was adopted.

8.2.2 Development of MGA on EA Suite platform

The information flow and pseudo-code of the MGA is shown in Fig. 8.3. Based on this, the MGA was implemented on the existing platform of EA Suite, and additional subroutines were included for this purpose. There were altogether seven newly developed subroutines as follows.

```
initialize the first population;
epoch = 1;
evaluate;
constraint handling;
identify the elite;
for epoch = 2: epoch_max do
        encode the population into binary strings;
        for i = 2:n_pop do
                tournament selection of 2 parents for recombination;
                recombination to reproduce an offspring;
        od
        convergence check for the offspring population;
        if convergence criterion satisfied
                keep elite and reinitialize a new population;
        else
                decode the offspring population from binary strings;
        fi
        evaluate;
        constraint handling;
        identify the elite;
od
```

Fig. 8.3. Pseudo-code of MGA

encode (ga_encode.m)

Encodes the real-valued individual into binary string according to the required resolution for bits and precision of representation.

tournament selection (t_selection.m)

Selects two parents through tournament selection based on the objective function value of the opponents for unconstrained problems, or based on both the objective function value and total constraint violation for constrained problems.

recombination (s_crossover.m)

Reproduces one offspring by single-point recombination.

recombination (u_crossover.m)

Reproduces one offspring by uniform recombination.

convergence check (convergence_check.m)

Checks the convergence of the genes across all the individuals in population according to the criterion of Senecal (2000).

reinitialize (reinitialize.m)

Reinitialize a new population of four if the convergence criterion is satisfied.

decode (ga_decode.m)

Decodes the binary individual back to the real-valued form according to the bits and precision of representation.

The existing platform of EA Suite provided the remaining operators and subroutines for implementation, such as evaluation, constraint handling, input/output utilities, etc.

8.2.3 Validation of developed MGA in EA Suite

Since the algorithm of MGA was developed with reference to the work of Krishnakumar (1989) and Senecal (2000), the test functions of their literature were used for validation purposes. There were two test functions in the former literature and one in the latter. Following the implementation details of the former, 25 runs were involved to acquire the "best-so-far-fitness". In the two papers, the results were only presented in

the form of graphs, and no actual numerical results were provided, so the results from the developed MGA were also compared in the form of graphs.

8.2.3.1 Krishnakumar's first test function

The first test function of Krishnakumar (1989) is in fact the well-known Sphere Model fu_1, but with the number of variables limited to three and narrower bounds of [-5.12, 5.12]. The published results from the paper, as well as the results from the developed MGA, are shown together in Fig. 8.4. It was found that the two graphs in Fig. 8.4 compare well.

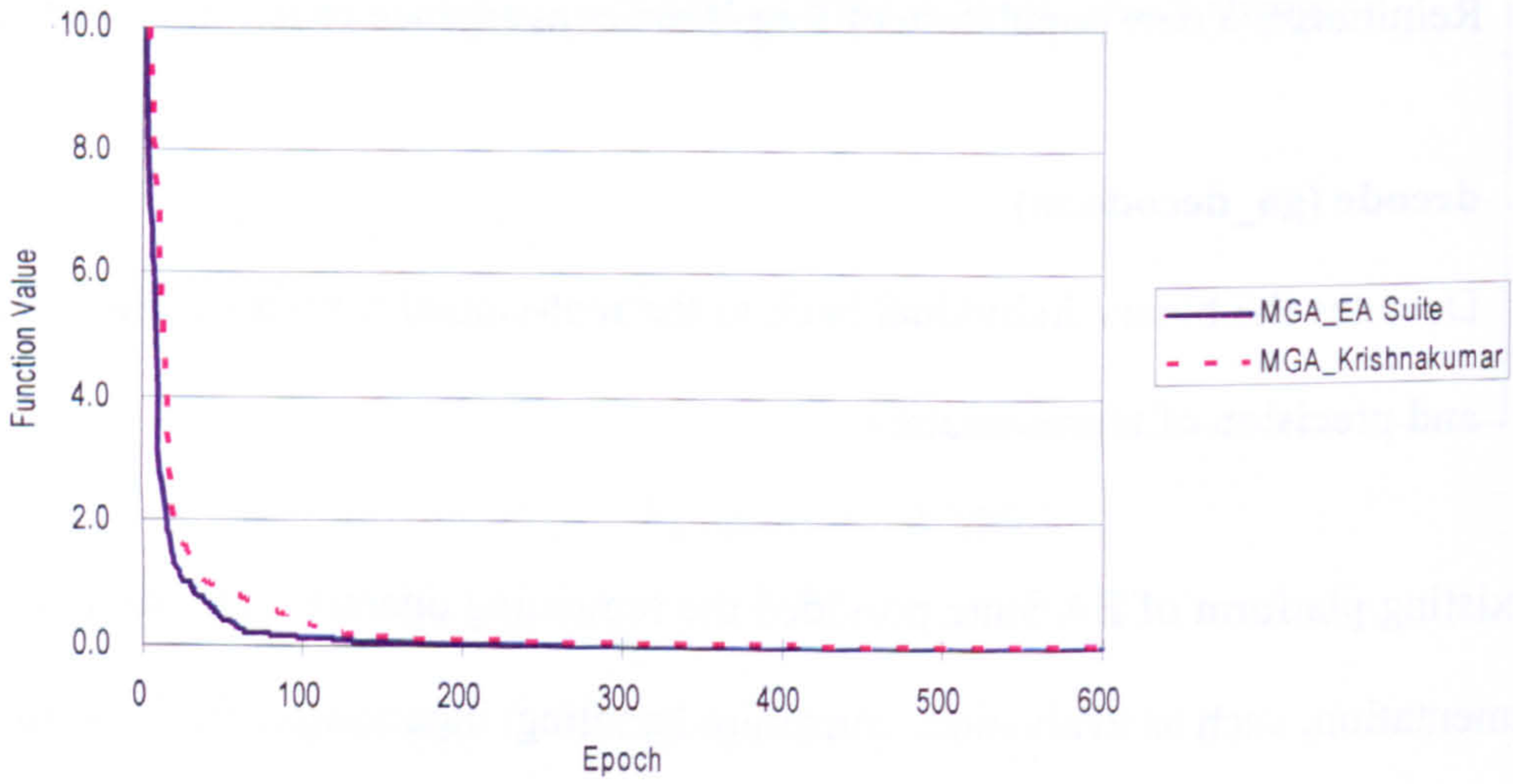


Fig. 8.4. Validation with Krishnakumar's first test function f_{u1} (n_{var}=3)

8.2.3.2 Krishnakumar's second test function

The second test function used by Krishnakumar is De Jong's stationary multimodal function (De Jong 1975), and it was added as f_{m6} in this EA Suite, as shown in Eq (A23) in Appendix I. This test function is non-convex with 25 local minima located at $x_i = a_{ij}$ ($i = 1, 2, ..., n_{pop}$; $j = 1, 2, ..., 25$). Since the coefficients a_{ij} were not stated in the

paper, they were all arbitrarily assigned to be 1 in this test. The published results in the paper, as well as the result from the developed MGA, are shown together in Fig. 8.5. Although the minimum values were not the same due to insufficient information of the coefficients a_{ij}, the evolution profile and convergence rate were similar, and the minimum was found after epoch 400 in both cases.

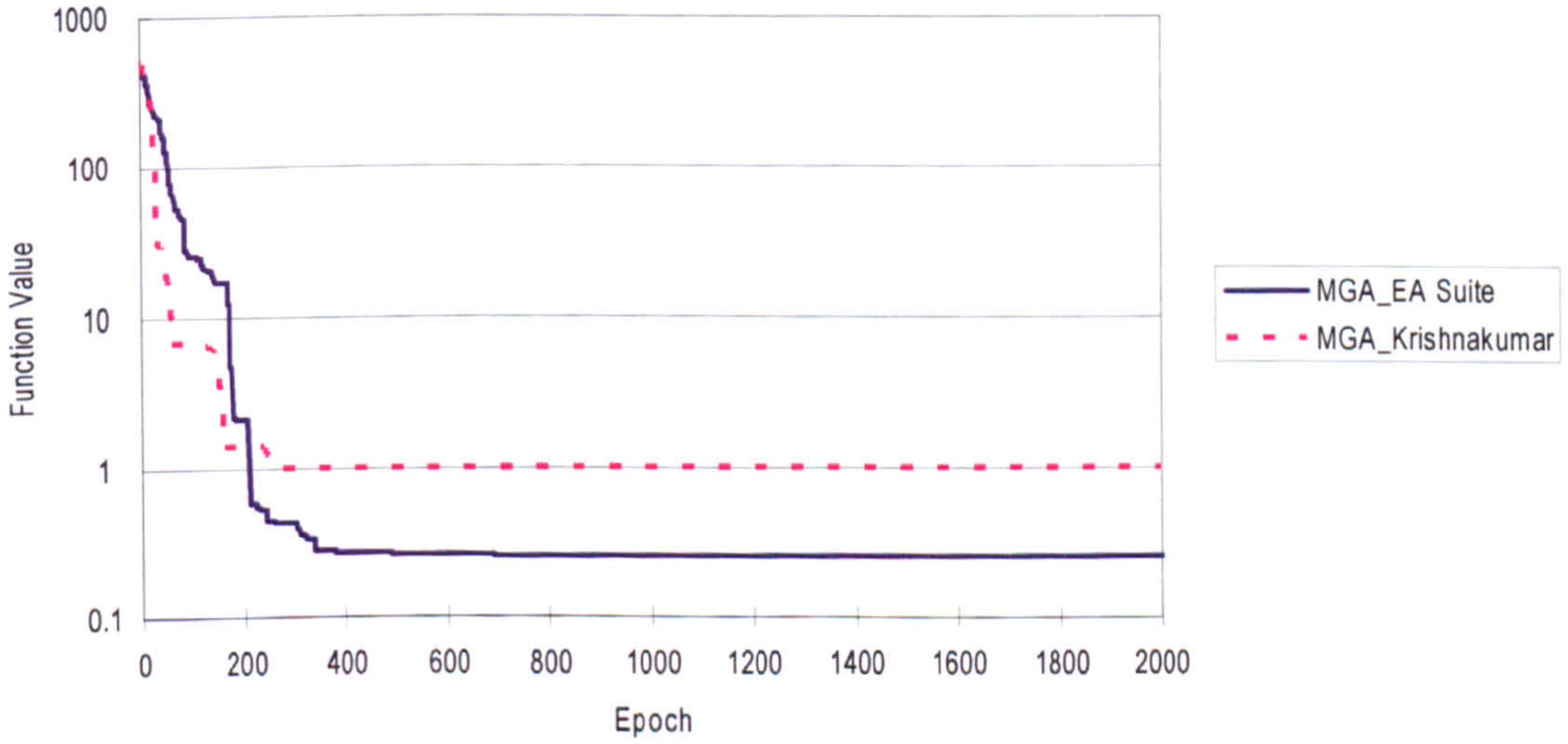


Fig. 8.5. Validation with Krishnakumar's second test function f_{m6}

8.2.3.3 Senecal's test function

The test function in Senecal (2000) was added as f_{m5} in this EA Suite. This is multi-modal with two variables x_1 and x_2, as shown in Eq (A22) in Appendix I. The published result in Senecal's paper and the result from the developed MGA are shown together in Fig. 8.6. The two graphs in Fig. 8.6 match well, with the evaluation function value approaching to the global maximum 1 from epoch 50.

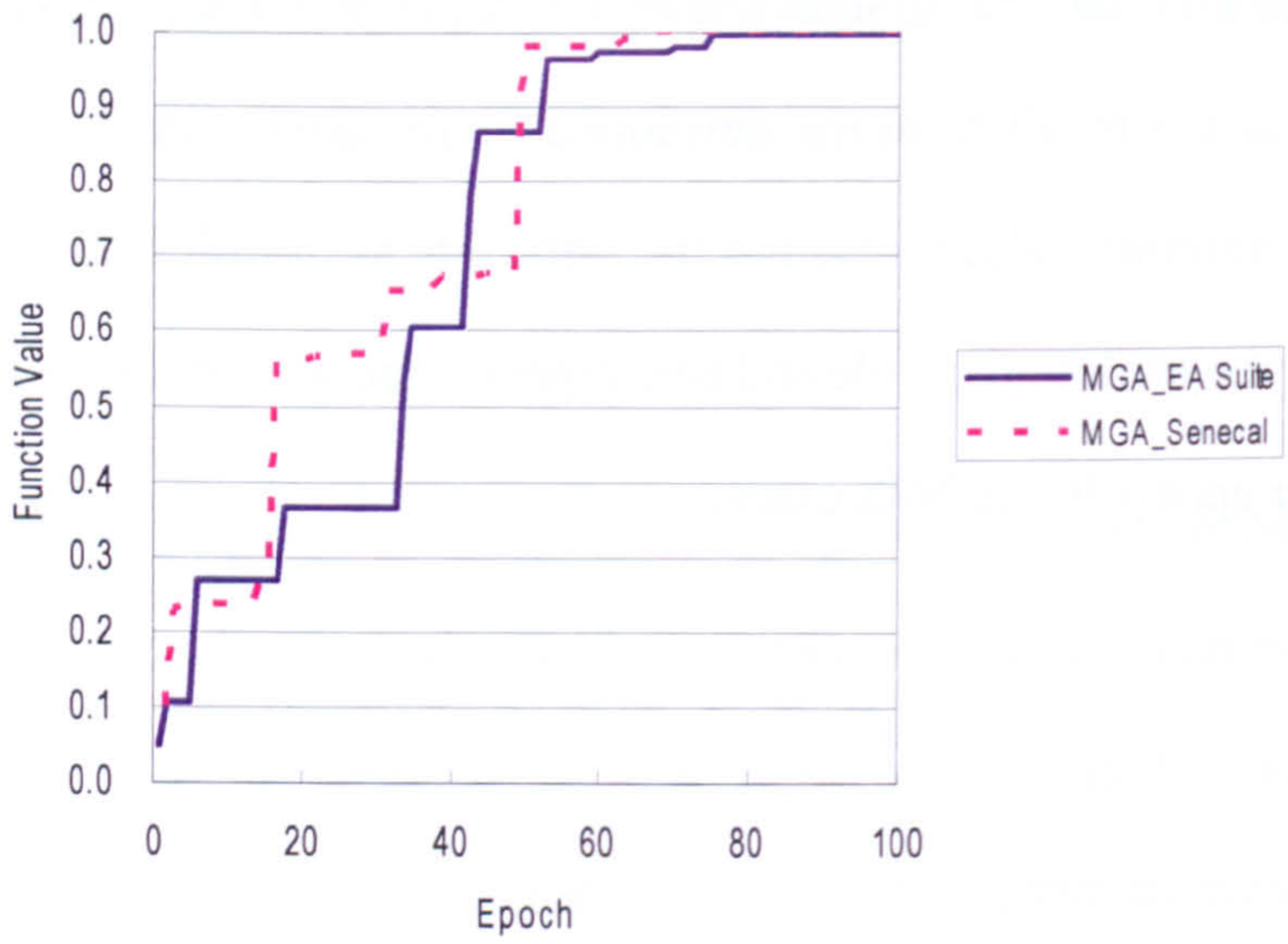


Fig. 8.6. Validation with Senecal's test function f_{m5}

8.2.4 Benchmarking through test functions from experimentation

Having validated the performance of the developed MGA on the EA Suite platform, it was used for benchmarking purposes against the proposed REA through the series of test functions involved in the experimentation described earlier in Chapter 6. With the same n_{pop} (= 5) for both the MGA and REA, a full run was implemented for the sixteen test functions, and the results presented in Table 8.1.

From Table 8.1 it can be seen that the REA had an overall performance better than MGA for almost all of the test functions, except f_{c7} and f_{m1}. However MGA could not handle five (f_{c1}, f_{c4}, f_{u1}, f_{u3} and f_{u4}) of the test functions and the results were far from the corresponding solutions, although the trend of continuing search was still found in some cases (f_{c4}, f_{u3} and f_{u4}). The operator of reinitialization with a small population (four plus the elite) provides too much diversity in the population, resulting in individuals scattered around the bounded search space.

Table 8.1. Benchmarking of REA with MGA

		MGA	**REA**
	XO	---	1 (Arithmetic)
	MU	---	2 (Cauchy)
	CH	1 (Infeasibility)	1 (Infeasibility)
	SE	---	2 (Tournament)
	n_{pop}	5	5
	$epoch_{max}$	1000*	1000*
Test Function	Best-known Solution	Mean Best (Std. Deviation)	Mean Best (Std. Deviation)
Constrained functions			
f_{c1} Floundas & P	-15	-9.60×10^{-1} (2.41)	-13 (9.98×10^{-1})
f_{c2} Himmelblau 11	-30665.5	-30290.28 (145.35)	-30567.74 (111.14)
f_{c3} Floundas & P	-6961.81	-4753.12 (2050.45)	-6845.92 (72.92)
f_{c4} Hock & S 113	24.31	333.57 (225.99)	25.47 (8.84)
f_{c5} Hock & S 100	680.63	442.60 (316.15)	681.62 (9.20×10^{-1})
f_{c6} Hock & S HX	7049.33	11201.76 (2674.49)	8514.78 (3452.18)
f_{c7} Maa & Shanblatt	0.75	0.7616 (2.12×10^{-2})	0.8037 (3.72×10^{-2})
Unconstrained unimodal functions			
f_{u1} Sphere Model	0	811.40 (450.16)	1.56×10^{-4} (7.18×10^{-5})
f_{u2} Schwefel 2.22	0	5.18 (1.80)	3.29×10^{-3} (8.58×10^{-4})
f_{u3} Schwefel 1.2	0	1920.44 (818.28)	2.29×10^{-1} (7.26×10^{-1})
f_{u4} Rosenbrock	0	28879.57 (48066.57)	39.43 (83.82)
f_{u5} Easom	-1	-0.76061 (3.71×10^{-1})	-0.95988 (1.98×10^{-1})
Unconstrained multimodal functions			
f_{m1} Schwefel 7	-4189.83	-3732.18 (183.15)	-3495.63 (248.23)
f_{m2} Rastrigin	0	17.77 (4.14)	9.55 (5.08)
f_{m3} Ackley	0	9.85 (1.97)	5.50×10^{-3} (1.32×10^{-3})
f_{m4} Griewank	0	8.56 (3.59)	1.53×10^{-1} (1.01×10^{-1})

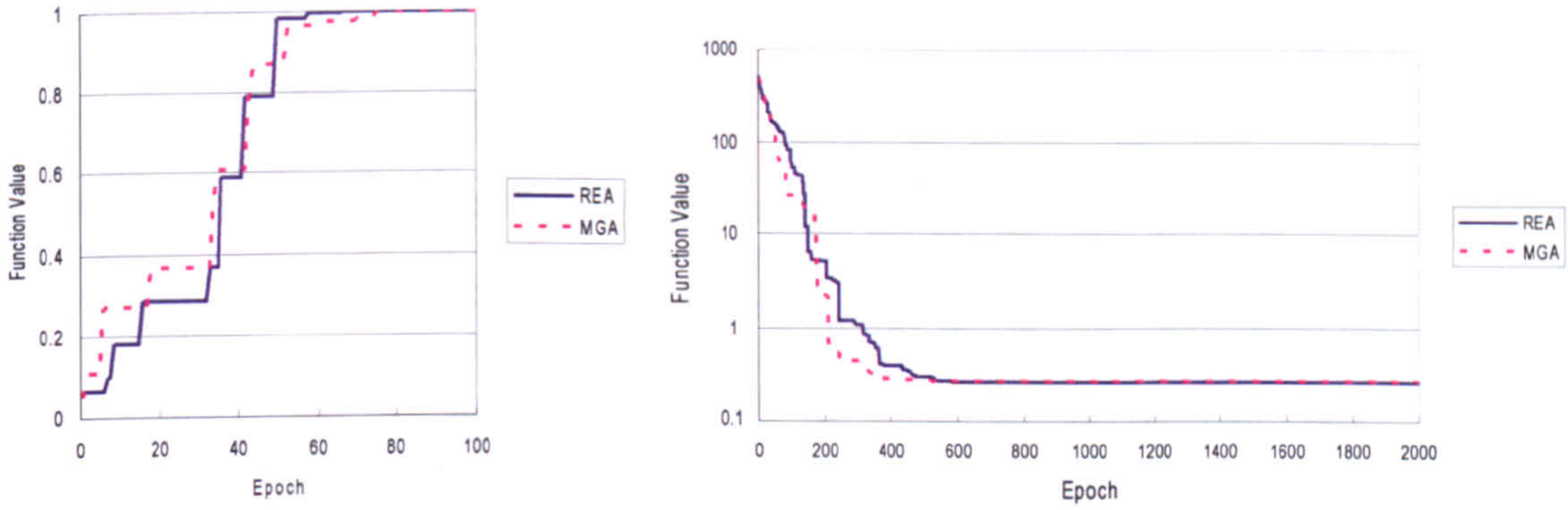
*Remark: For f_{c6}/f_{u4} and f_{m1}/f_{m2}, $epoch_{max}$ were 3000 and 2000 respectively.

In addition, MGA could not determine the optimum for f_{c1} at all. This was because f_{c1} has a local minimum at $x_i = 0$ (for i = 1, 2, ..., 13), with both the objective function and constraint values equal to zero. This feasible individual would be easily chosen to be the elite. Whenever the reinitialization operator performed its role after the convergence of the micro-population, this "elite" individual would be naturally selected, and the newly reinitialized individuals would inevitably have a finite constraint violation. Therefore this "elite" individual was carried forward continually, and become the final "optimal" solution determined by MGA.

8.2.5 Conclusion

Although MGA could provide satisfactory results for those chosen test functions as published in Krishnakumar (1989) and Senecal (2000), it did not have an overall satisfactory performance for other test functions used in the current study. In addition, the MGA could not handle five of the test functions at all because of the effects of the reinitialization operator and fine-tuning of the elite could not be performed.

On the other hand, the comparison showed that the REA ($n_{pop} = 5$) could have overall a satisfactory performance for the series of test functions. For test functions f_{m5} and f_{m6} used by Senecal (2000) and Krishnakumar (1989) respectively, the results of the REA ($n_{pop} = 5$) were comparable to those of the MGA in the EA Suite, as shown in Fig. 8.7.



(a) Test function f_{m5} (b) Test function f_{m6}

Fig. 8.7. Performance benchmarking of REA (n_{pop} = 5) with MGA for f_{m5} and f_{m6}

8.3 Contribution of Constraint Handling Techniques

8.3.1 Degree of contribution of constraint handling techniques in different EA paradigms

In recent years, one of the interests of EA researchers was to develop appropriate and efficient constraint handling techniques for constrained problems (Coello 2000a, 2000b, 2002, Deb 2000, Runarsson and Yao 2000, Wright and Farmani 2001, Michalewicz and Fogel 2004). The constraint functions would be linear or nonlinear, equality or inequality in nature. The penalty-based methods have a long history of applications due to their simplicity. As discussed *in Section 2.5.1, the penalty* parameters are usually problem-specific and not commonly transferable. To face this challenge, different improved methods of constraint handling techniques have emerged, not just to facilitate the efficiency of global search, but more important, to acquire better and more reliable global solutions. In the paradigm of GA, apart from the core operators like recombination and selection, constraint handling has become another major operator (Wright 1996), even more important than the background operators like mutation and migration. In the paradigms of evolutionary programming and evolution strategy, no

in-depth study has been conducted on the effectiveness of constraint handling techniques. It is generally supposed that available constraint handling methods would be paradigm-independent and their performance and degree of importance would be similar to those as found in GA.

Deb (2000) and Coello (2002) carried out thorough comparative studies of different constraint handling techniques in the context of EA. Deb compared the performance of the constraint handling method from Powell and Skolnick (1993) and his proposed constraint handling method of tournament selection operator for real-valued GA (TS-R) for seven constrained test functions together with the well-studied welded beam design problem (Reklaitis *et al.* 1983). It was found that TS-R method was more effective with the feature of no problem-specific penalty parameter. The detail of TS-R method is described earlier in Section 2.5.2.2.

On the other hand, Coello used a common test function and two engineering problems – a similar welded beam design problem (Rao 1996) and the pressure vessel design problem (Kannan and Kramer 1994) for comparative study. In Coello's discussion, his proposed non-dominance method was more superior than three types of constraint handling methods, including the traditional methods (Himmelblau 1972, Siddall 1972, Ragsdell and Philips 1976, Kannan and Kramer 1994 and Sandgren 1988); special methods for GA (Gen and Cheng 1997, Deb 1991 and Deb 1997); and penalty-based methods (Homaifar *et al.* 1994, Joines and Houck 1994, Michalewicz and Attia 1994, Hadj-Alouane and Bean 1997). The features and implementation of non-dominance method is described earlier in Section 2.5.2.4.

From the works of Deb and Coello, effective constraint handling techniques were shown to help the evolutionary algorithm in finding the global or near-optimum.

However all their comparative studies and proposed improved methods were developed on the basis of GA only, not evolutionary programming or evolution strategy. Although a real-parameter GA was used by Deb, it was different from evolution strategy commonly handling real-valued variables. Since in the real-parameter GA, mutation is just a background operator with low probability of execution. On the contrary, mutation is one of the core operators in the paradigm of evolution strategy, even more significant than recombination as concluded in Section 7.5.2. Therefore a series of comparative studies were carried out to benchmark the proposed REA with the effective TS-R method and non-dominance method advocated by Deb (2000) and Coello (2002) respectively.

8.3.1.1 Benchmarking with TS-R method

In TS-R method (Deb 2000), apart from using mutation to maintain the diversity of population, the niching method was also applied for the same purpose and implemented during tournament selection. Niching started from calculating the normalized Euclidean distance between two opponents, if the value was less than a preset critical distance, the two opponents would be compared based on the objective function value, otherwise no tournament would be carried out thus diversity could be maintained. In Deb (2000) demonstrated that good results for his test problems were achieved in the presence of niching without mutation at a medium epoch of termination. Better results of the test problems were obtained by involving both niching and mutation, the epochs of termination were also much larger, of the order of thousands. Therefore, the performance of the TS-R method only with niching was used for this comparative study. The results of the TS-R method from Deb (2000) and the REA are shown together in Table 8.2. The items of comparison and format of presentation were based on Deb's

paper. There were three major areas of comparison:

a. Deviation percentage from the best-known solution;

b. Statistical data, including the best, median or worst solution of 50 runs; and

c. Number of evaluation function calls.

Table 8.2. Benchmarking of REA with TS-R method for test problems of Deb (2000)

Problem (Best -known Solution)	Method	Number of individuals						Best	Median	Worst	No. of Function Calls
		Deviation Percentage from Best-known Solution									
		≤ 1%	≤ 2%	≤ 5%	≤ 10%	≤ 20%	≤ 50%				
f_{c1}	TS-R	*47*	*47*	*47*	*47*	*50*	*50*	*-15*	*-15*	*-13*	*65130*
Floundas & P	REA-1	26	26	26	44	50	50	-15	-15	-12	9001
(-15)	REA-2	26	26	26	42	50	50	-15	-15	-13	9001
	REA-3	28	28	28	42	50	50	-15	-15	-13	9001
f_{c2}	TS-R	*17*	*44*	*50*	*50*	*50*	*50*	*-30646.47*	*-30279.74*	*-29794.44*	*50050*
Himmelblau	REA-1	50	50	50	50	50	50	-30666.34	-30666.30	-30418.29	9001
(-30665.5)	REA-2	50	50	50	50	50	50	-30666.34	-30666.30	-30464.78	9001
	REA-3	50	50	50	50	50	50	-30666.34	-30666.31	-30486.95	9001
f_{c4}	TS-R	*0*	*0*	*9*	*25*	*36*	*45*	*24.87747*	*26.73401*	*50.40042*	*100100*
Hock & S 113	REA-1	1	2	19	45	48	50	24.34917	25.73943	29.47076	9001
(24.31)	REA-2	3	5	18	34	49	50	24.44980	25.99148	29.29048	9001
	REA-3	1	2	19	40	49	50	24.54892	25.82319	30.95845	9001
f_{c5}	TS-R	*50*	*50*	*50*	*50*	*50*	*50*	*680.6594*	*681.5256*	*687.1886*	*70070*
Hock & S 100	REA-1	50	50	50	50	50	50	680.6473	680.8324	681.2939	9001
(680.63)	REA-2	50	50	50	50	50	50	680.6585	680.8496	681.3875	9001
	REA-3	50	50	50	50	50	50	680.6711	680.8537	681.5510	9001
f_{c6}	TS-R	*3*	*5*	*7*	*14*	*29*	*47*	*7065.742*	*8274.830*	*10925.17*	*80080*
Hock & S HX	REA-1	8	17	26	36	45	50	7076.204	7368.519	10088.59	27001
(7049.33)	REA-2	3	10	23	39	44	50	7069.274	7437.398	10225.97	27001
	REA-3	11	13	23	31	41	47	7094.143	7558.382	13651.48	27001
f_{c8}	TS-R	*29*	*31*	*31*	*32*	*33*	*39*	*13.59085*	*13.61673*	*117.0297*	*2550*
Deb 1	REA-1	43	43	43	43	43	45	13.58409	13.58627	78.61142	1801
(13.59085)	REA-2	38	38	38	38	38	38	13.58408	13.58789	268.9163	1801
	REA-3	40	40	41	41	41	42	13.58428	13.58675	255.5800	1801
f_{c9}	TS-R	*24*	*24*	*27*	*32*	*47*	*50*	*-1.91410*	*-1.85504*	*-1.30643*	*20050*
Hock & S 85	REA-1	5	11	21	30	50	50	-2.17812	-1.96749	-1.72091	9001
(-1.91460)	REA-2	3	8	18	36	50	50	-2.17532	-2.02068	-1.72948	9001
	REA-3	3	5	19	33	50	50	-2.17577	-2.03984	-1.78683	9001
f_{c10}	TS-R	*28*	*36*	*44*	*48*	*50*	*50*	*2.38119*	*2.39289*	*2.64583*	*40080*
W. Beam 1	REA-1	26	33	46	49	49	50	2.38086	2.40424	2.97595	29001
(2.38116)	REA-2	20	30	47	49	50	50	2.38022	2.41644	2.67375	29001
	REA-3	31	39	46	48	49	50	2.37958	2.39383	2.89087	29001

Remark: REA-1, REA-2 and REA-3 refer to the REA using the constraint handling operator of infeasibility discrimination (CH=1), stochastic ranking (CH=2) and dynamic penalty (CH=3) respectively.

For the TS-R method, the total number of evaluation function calls n_{eval} was determined by Eq (8.1) below.

$$n_{eval} = n_{pop} (epoch_{max} + 1) \tag{8.1}$$

Since initialization was not counted as a nominal epoch, an additional evaluation was required for the population after initialization in the TS-R method. On the other hand, for the REA, the total number of evaluation function calls was determined according to Eq (5.32) in Chapter 5, based on elitism from the second epoch.

Better results for the REA using any kind of constraint handling operator against that of the TS-R method, are shown shaded in Table 8.2. It was found that the REA had overall a better performance for all the test problems in all the three areas of comparison, including deviation percentage from to the best-known solution, statistical outcomes and number of function calls. In addition, the REA found better results than the corresponding best-known results for the three test functions f_{c2}, f_{c8}, f_{c9} and the welded beam test problem f_{c10}, as indicated by the dotted rectangles ⬚. The better performance of the REA was not due to the contribution of the constraint handling operators, since the results across the three kinds of constraint handling techniques were similar. The major contribution to the search effectiveness was more related to the effective combination of the core operators of the REA.

8.3.1.2 Benchmarking with non-dominance method

The non-dominance method is a constraint handling method developed by Coello (2002). Since Coello (2002) concluded that his non-dominance method had overall better performance than the other kinds of methods (traditional methods, special methods for GA and penalty-based methods), so the comparative study was concentrated on

performances between the non-dominance method and the REA using different constraint handling operators. The benchmarking results are consolidated in Table 8.3. The format of this comparative study is based on that of Coello (2002). Thirty runs (instead of the default fifty runs) were used in order to have same comparison basis with Coello's work. In addition to the statistical results of 30 runs, the number of evaluation function calls is also presented in Table 8.3 for benchmarking purposes. Where the result of the REA was better than that of the non-dominance method or the best-known solution, it is shown shaded in the table.

Table 8.3. Benchmarking of REA with non-dominance method for test problems of Coello (2002)

Problem (Best -known Solution)	Method	Best	Mean	Worst	Std. Dev.	No. of Function Calls
f_{c2}	Non-dominance	*-31005.7966*	*-30862.8735*	*-30721.0418*	*73.240*	*5000*
Himmelblau	REA-1	-30666.34	-30580.46	-30343.03	100.024	2701
(-30665.5)	REA-2	-30666.31	-30543.41	-29828.72	205.575	2701
	REA-3	-30666.29	-30558.30	-30291.29	113.974	2701
f_{c11}	Non-dominance	*1.7483*	*1.7720*	*1.7858*	*0.01122*	*5000*
W. Beam 2	REA-1	1.7317	2.1536	4.1726	0.49719	4501
(1.74830941)	REA-2	1.7348	2.1879	3.3753	0.46049	4501
	REA-3	1.7610	2.1343	3.6590	0.42524	4501
f_{c12}	Non-dominance	*6069.3267*	*6263.7925*	*6403.4500*	*97.9445*	*50000*
Press. Vessel	REA-1	6053.6950	6190.0640	6608.0670	174.1472	18001
(6288.7445)	REA-2	6053.8410	6198.5150	6608.0310	183.3954	18001
	REA-3	6053.6080	6202.8340	6607.8680	193.3922	18001

Remark: REA-1, REA-2 and REA-3 refer to the REA using the constraint handling operator of infeasibility discrimination (CH=1), stochastic ranking (CH=2) and dynamic penalty (CH=3) respectively.

For the Pressure Vessel Design Problem f_{c12}, Table 8.3 shows that the REA with any kind of constraint handling operator gave better overall results, with respect to both best and mean results, as well as the number of evaluation function calls. For the test function f_{c2}, the best and mean results of the REA were not as good as those of the non-dominance method, but the REA with all the three kinds of constraint handling

operators could search for better minimum values against the best-known solution, like the non-dominance method did. The number of evaluation function calls of REA was only about half of that of the non-dominance method. Therefore these two methods were considered comparable for the test function f_{c2}. For the Welded Beam Design Problem 2 f_{c11}, the mean values in 30 runs of the REA were worse than that of the non-dominance method, however the REA with the constraint handling operators of infeasibility discrimination and stochastic ranking could find better minimum values than the best-known solution as determined by the non-dominance method. In addition, the number of evaluation function calls of the REA was slightly less. As a result, the overall performance of the REA for the Welded Beam Design Problem 2 f_{c11} was also considered comparable to that of the non-dominance method.

As a whole, the performance of different constraint handling operators under the REA was comparable to that of the non-dominance constraint handling method, particularly better for the practical Pressure Vessel Design Problem f_{c12}.

8.3.1.3 Conclusion

From the benchmarking studies with both the Deb's TS-R method and Coello's non-dominance method, it was found that the REA could generally provide better solutions with comparatively less evaluation function calls. The better performance was not related to the effectiveness of the constraint handling operators, since the results across the three kinds of the involved constraint handling operators were similar. Therefore the effectiveness and efficiency of the REA could result from the synergetic combination of the core operators of mutation, selection and recombination. Since the REA was developed upon the paradigms of evolutionary programming and evolution strategy, the contribution of the constraint handling techniques were not as significant as

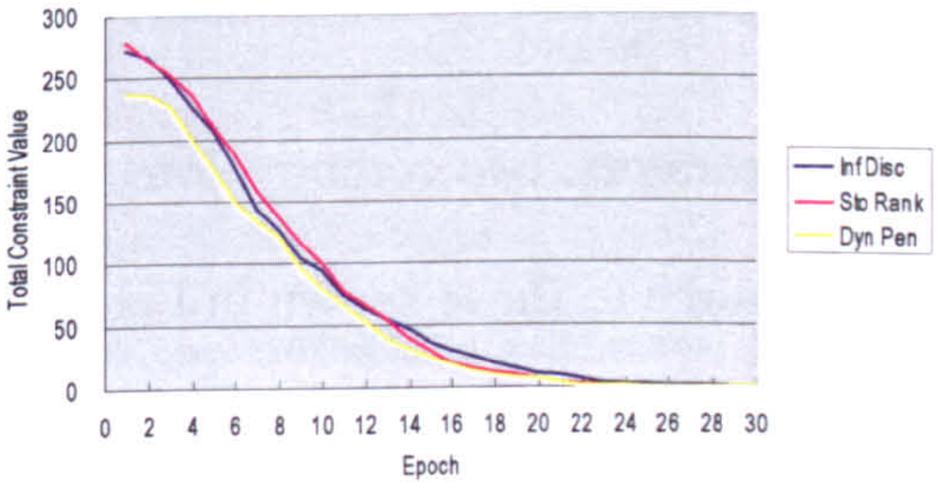
under the paradigm of GA. Further study about the limited contribution of constraint handling techniques will be discussed in the ensuing Section 8.3.2.

8.3.2 Qualitative analysis of constraint handling operators in EA Suite

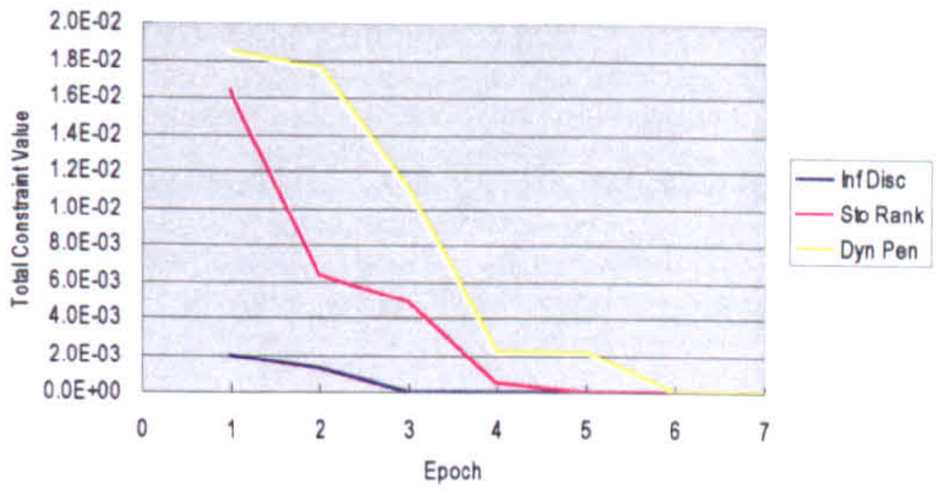
8.3.2.1 Effect of constraint handling operators on constraint profiles

In this part of study, an in-depth analysis was carried out on the profiles of total constraint violation of the elite individual along the epoch of evolution. The profiles were acquired from the full runs of the constrained test functions, and they were used to support the proposition that the contribution from constraint handling operators in the REA was limited. From the experimental results of each constrained test function in Section 7.4, the changing profiles of the average total constraint value against epoch in the 50 runs were plotted. Fig 8.8 consolidates these profiles for the constrained test functions f_{c1} to f_{c7}. Each constraint handling operator was used with the core EA operators of Cauchy deterministic mutation, tournament selection and arithmetic recombination of the REA.

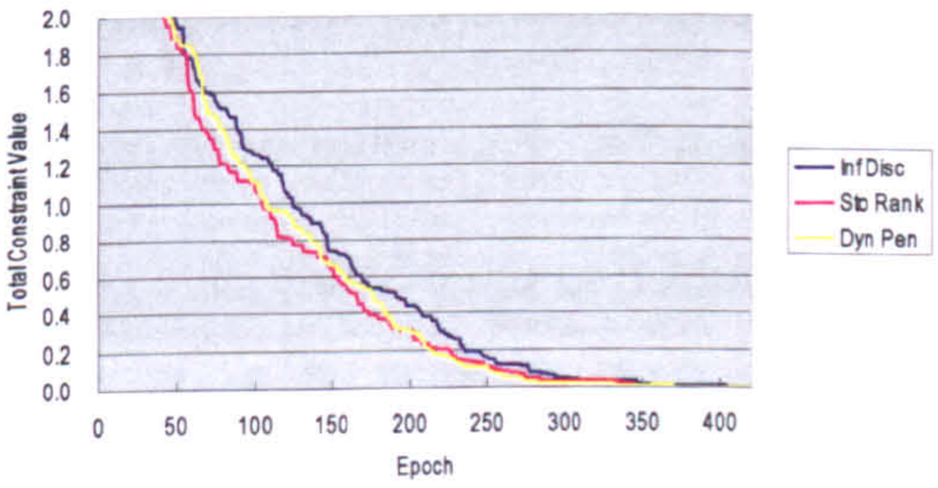
From Fig. 8.8, different constraint handling operators had a similar performance under the REA. In general, the total constraint value became negligible at about 0.2% to 4% of the typical epoch of termination of 1000 (except the test function f_{c3}). This showed that the elite search could be confined within the feasible region very shortly after starting the EA. The continuing global search would then mainly rely on the core EA operators, and the constraint handling operator would become a background operator used to distinguish the feasible and infeasible individuals in the ongoing epochs.



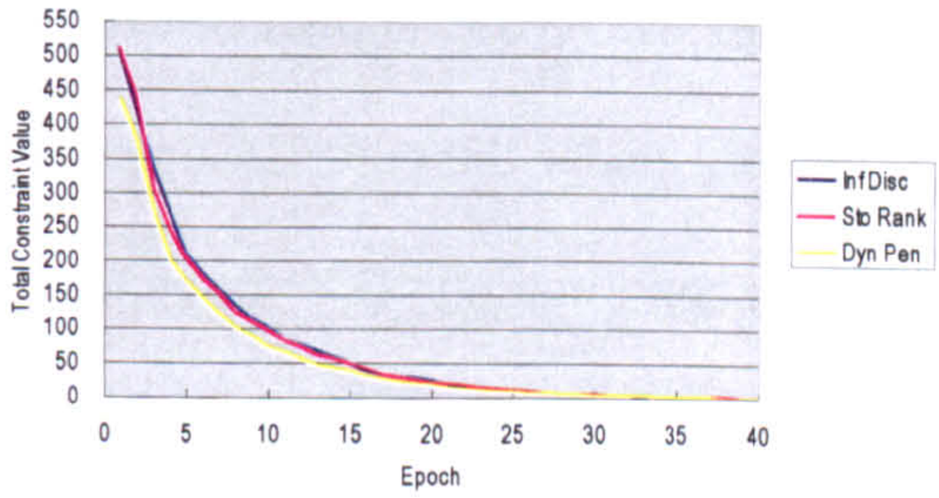
(a) Test function f_{c1}



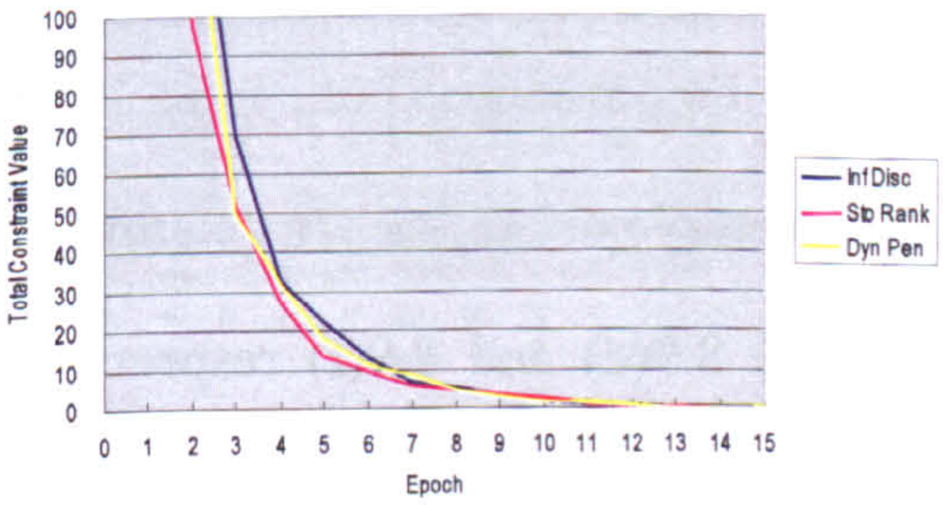
(b) Test function f_{c2}



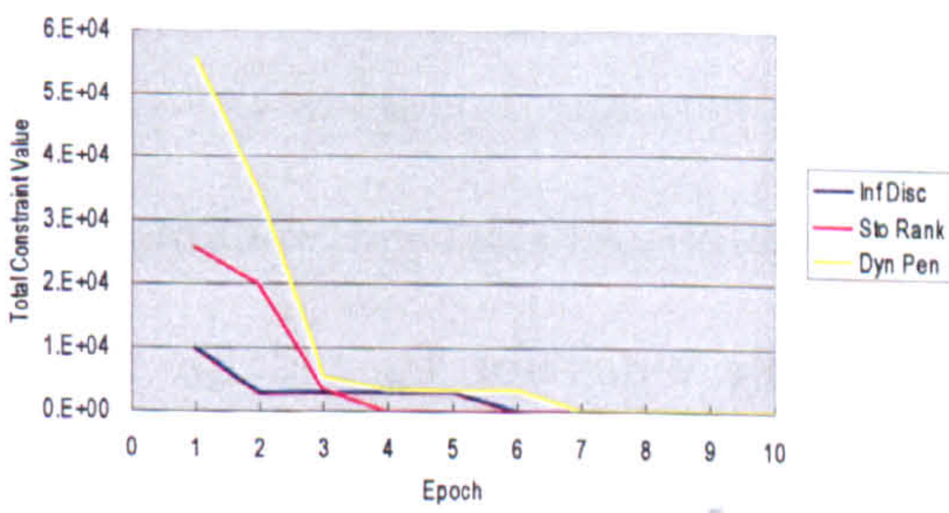
(c) Test function f_{c3}



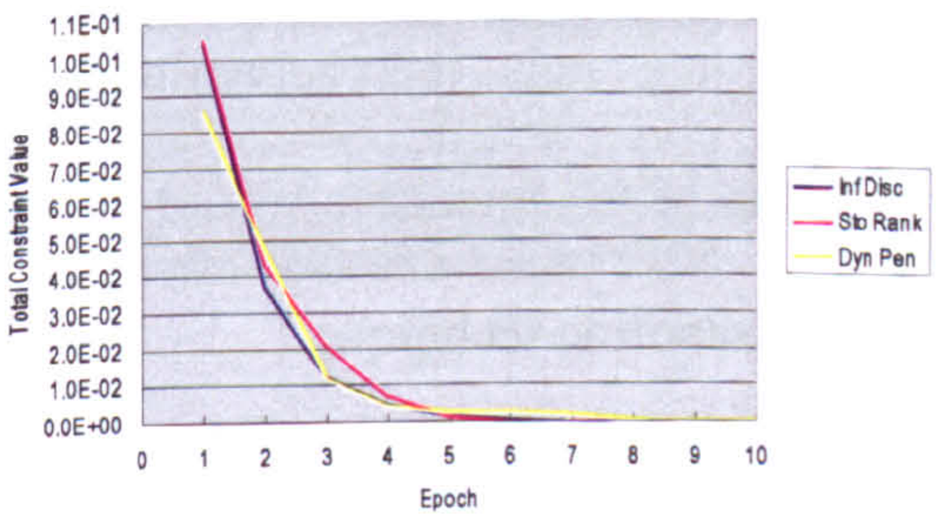
(d) Test function f_{c4}



(e) Test Function f_{c5}



(f) Test Function f_{c6}



(g) Test Function f_{c7}

Fig. 8.8. Constraint profiles of different constraint handling operators by using REA

Remarks:

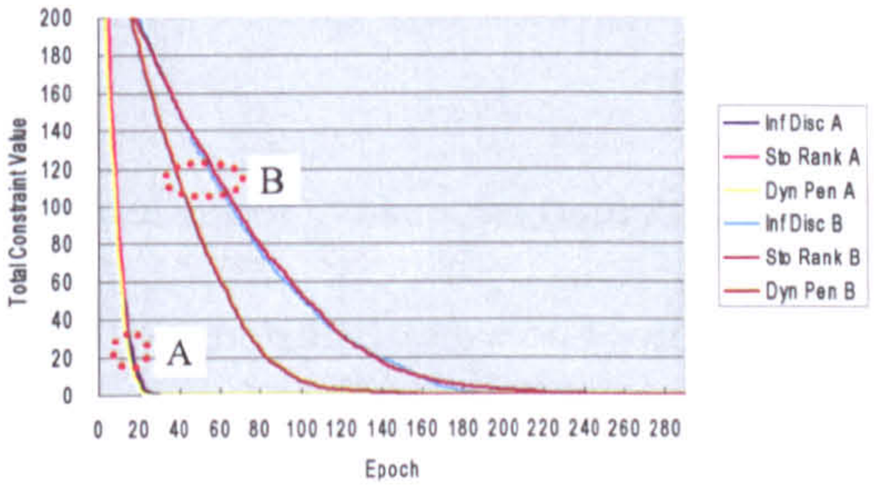
a. Abbreviation:
   Inf Disc: infeasibility discrimination; Sto Rank: stochastic ranking; Dyn Pen: dynamic penalty
b. The range of the ordinate of each figure was adjusted for easy observation of the changing performance of the corresponding test function.

8.3.2.2 Effect of core EA operators on constraint profiles

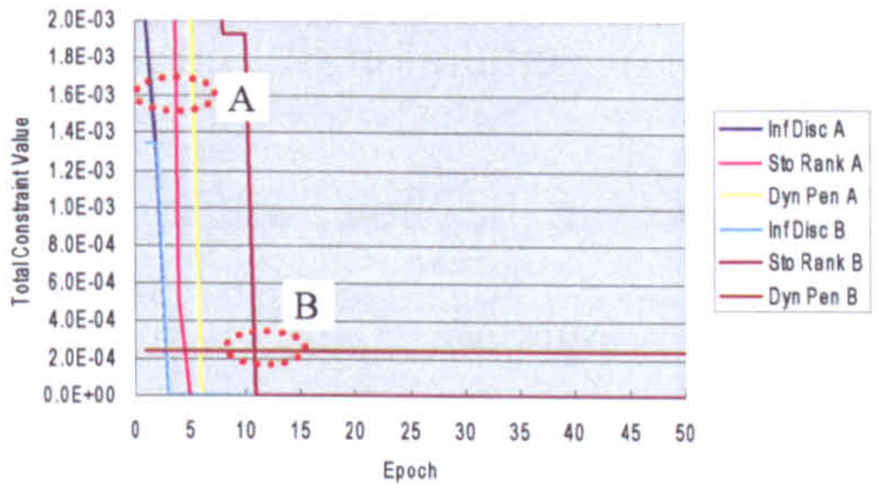
In order to study the influence of the core EA operators, two combinations were used. Combination "A" was the REA with identical results to those shown in Fig. 8.8. Combination "B" consisted of Gaussian deterministic mutation, ranking selection and no recombination. Both combinations used a population of ten.

From the results shown in Fig. 8.9, it can be seen that the constraint handling operator used under the core EA operators in combination "A" had better performance than that in combination "B". In general, the former could find the feasible elite earlier than the latter. Combination "B" had a finite constraint value at larger epochs, even up to the epoch of termination for the test functions f_{c2} and f_{c6}.

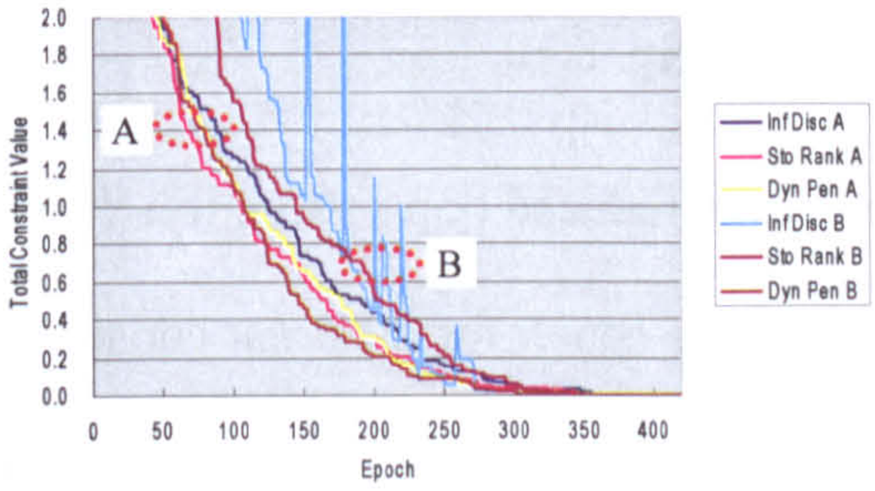
On the other hand, if combination "B" was used, the performance of different constraint handling operators deviated significantly. For instance, both infeasibility discrimination and stochastic ranking showed unsatisfactory performance in determining the feasible elites for f_{c4} and f_{c6}, as shown in Figs. 8.9(d) and 8.9(f) respectively. However the dynamic penalty had a relatively stable performance for different test functions in general. Therefore with the mix of EA operators in combination "B", effective constraint handling operators might not be able to exercise their capability in an elite search. As a result, the choice of the combination of EA operators would in turn affect the effectiveness and efficiency of the constraint handling technique.



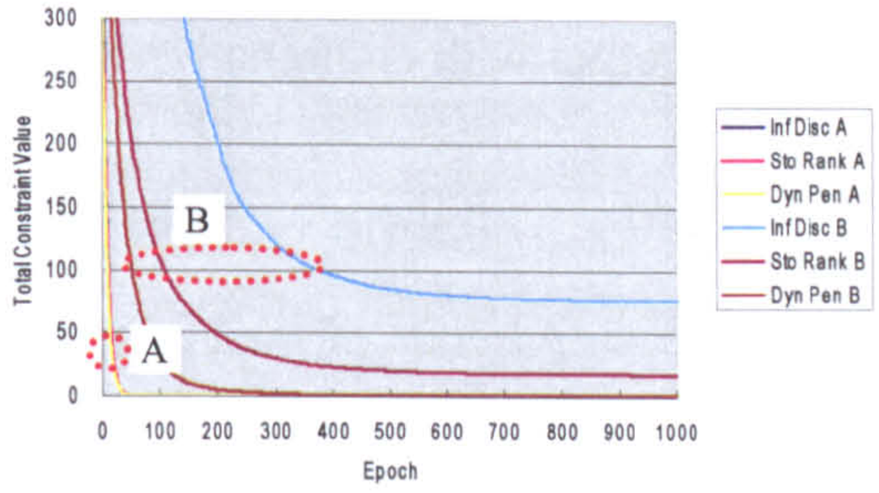
(a) Test function f_{c1}



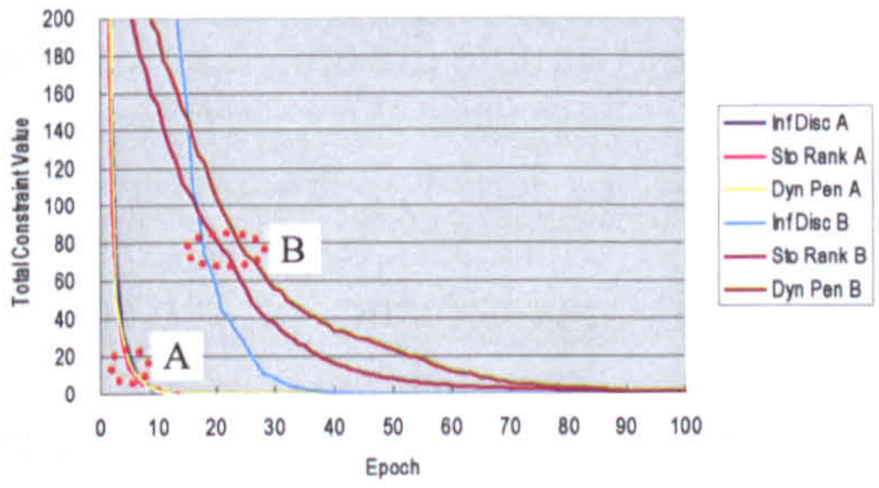
(b) Test function f_{c2}



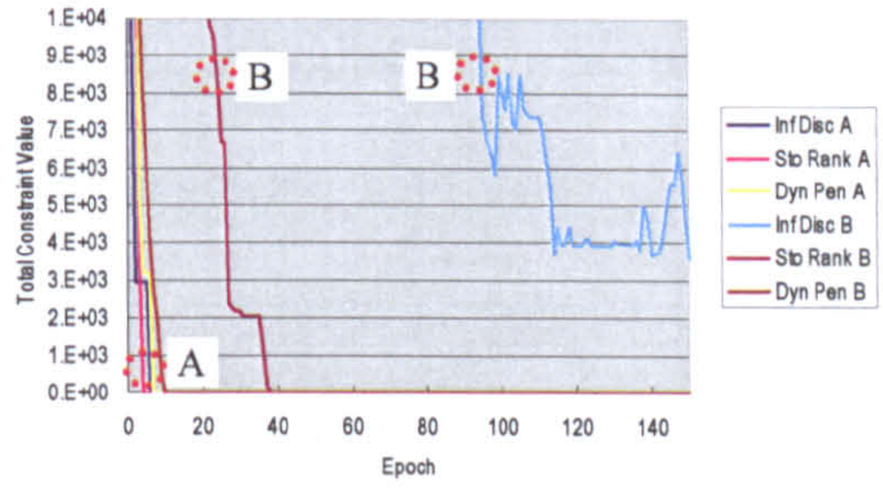
(c) Test function f_{c3}



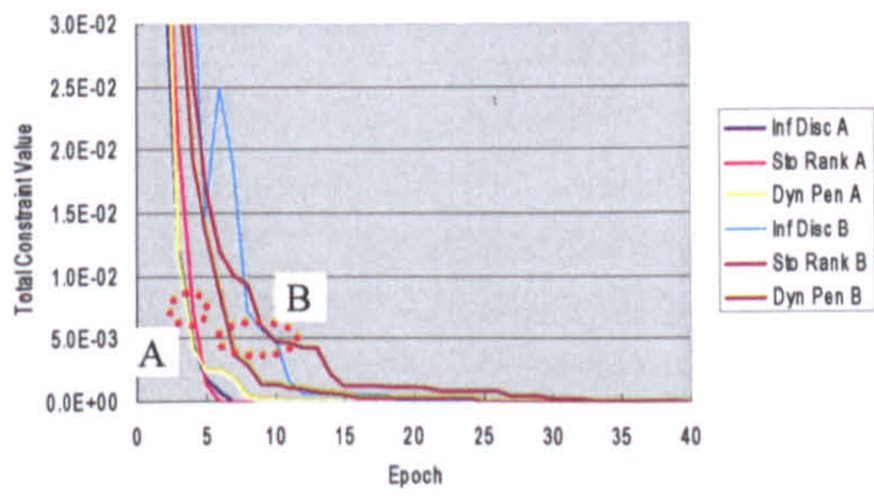
(d) Test function f_{c4}



(e) Test Function f_{c5}



(f) Test Function f_{c6}



(g) Test Function f_{c7}

Fig. 8.9. Constraint profiles of combinations A and B of EA operators at n_{pop}=10

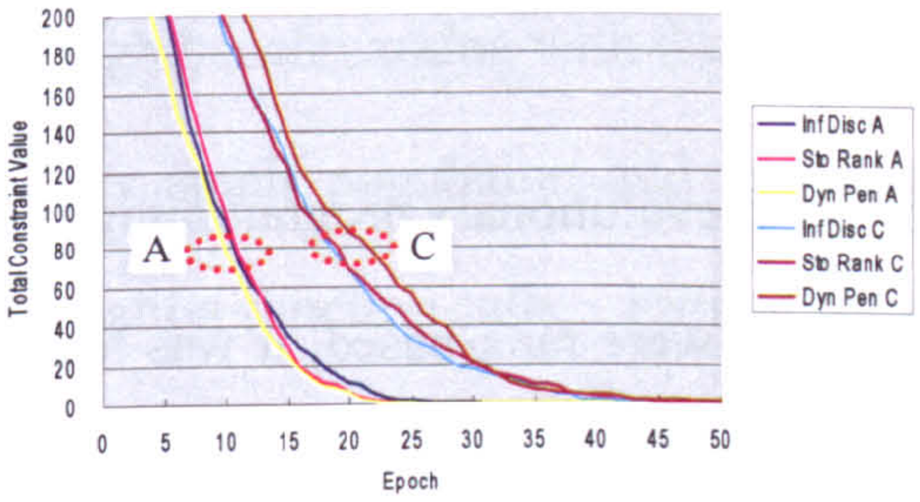
Remarks:

a. Abbreviation:
   Inf Disc: infeasibility discrimination; Sto Rank: stochastic ranking; Dyn Pen: dynamic penalty
b. The range of the ordinate of each figure was adjusted for easy observation of the changing performance of the corresponding test function.

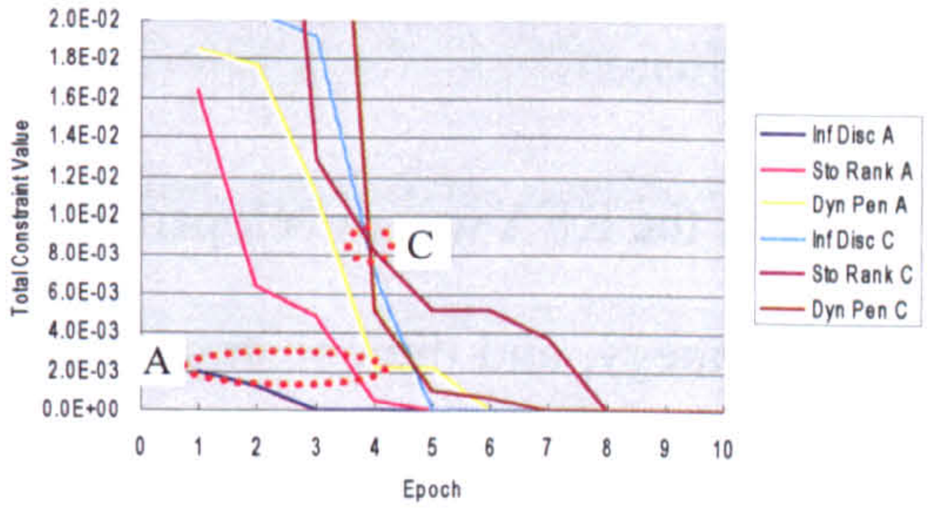
8.3.2.3 Effect of population size on constraint profiles

Apart from the choice of the core EA operators, it is also necessary to understand the effect of population size. Owing to this, combination "C" was designed, with the same operators of mutation, selection and recombination as in the REA, but the population was reduced to five. The results of combination "C" for the different test functions, together with combination "A", are plotted in Fig. 8.10.

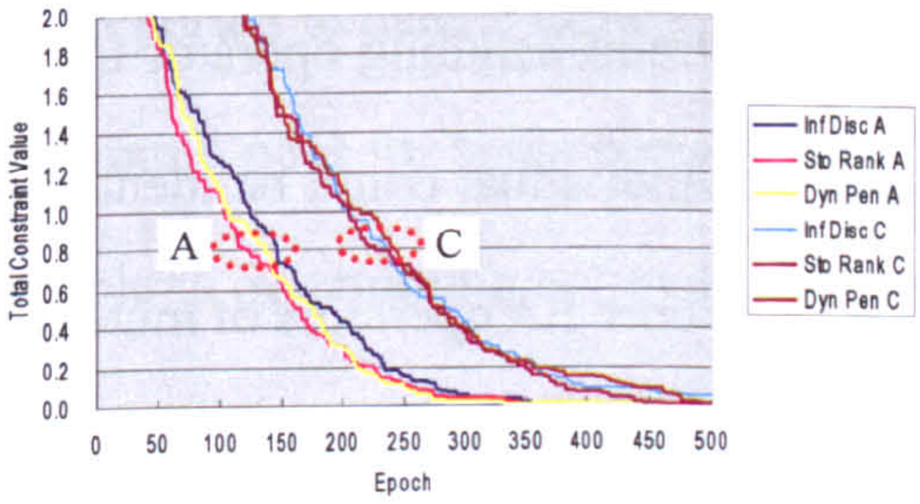
Generally with the same core EA operators at the reduced population, it is found that the decreasing trends of total constraint value against epoch for different constraint handling operators were similar, as demonstrated in the test functions f_{c1}, f_{c3}, f_{c4} and f_{c5}. On the other hand, the profiles at the reduced population were shifted to the right as compared to those at the default population of ten, with about double the epoch before the total constraint value died out, again obvious for f_{c1}, f_{c3}, f_{c4} and f_{c5}. In fact for the remaining three constrained test functions, there were similar performance profile but the changing trend was not so apparent, since the elite still with a finite constraint value was at low epoch, from 8 to 25 only. Therefore the effect of different constraint handling operators was generally similar under the same population size.



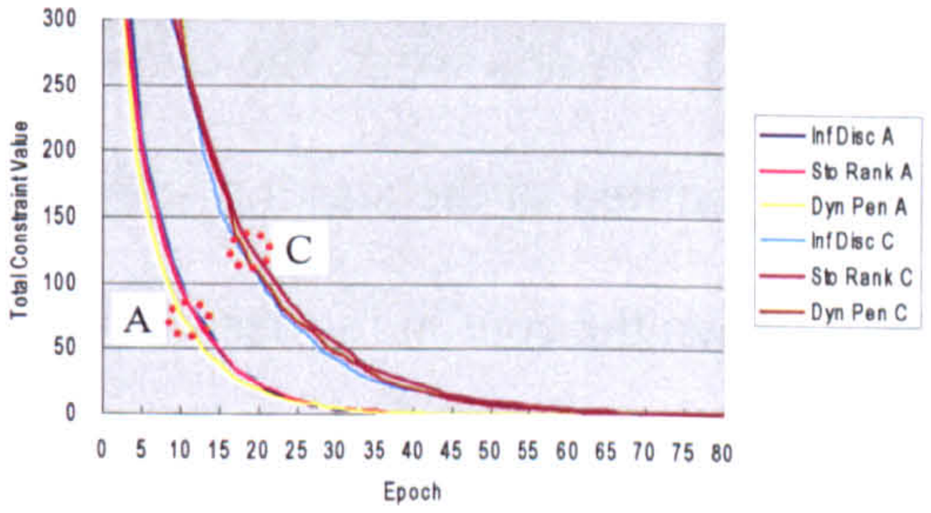
(a) Test function f_{c1}



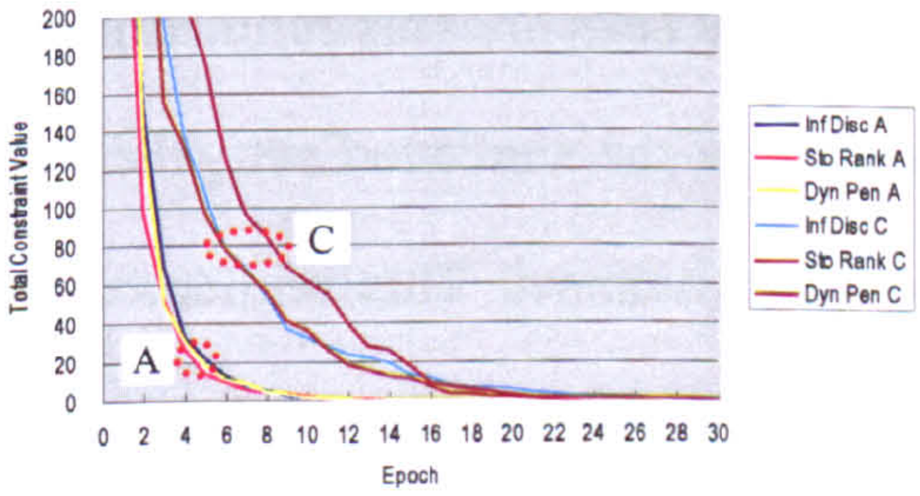
(b) Test function f_{c2}



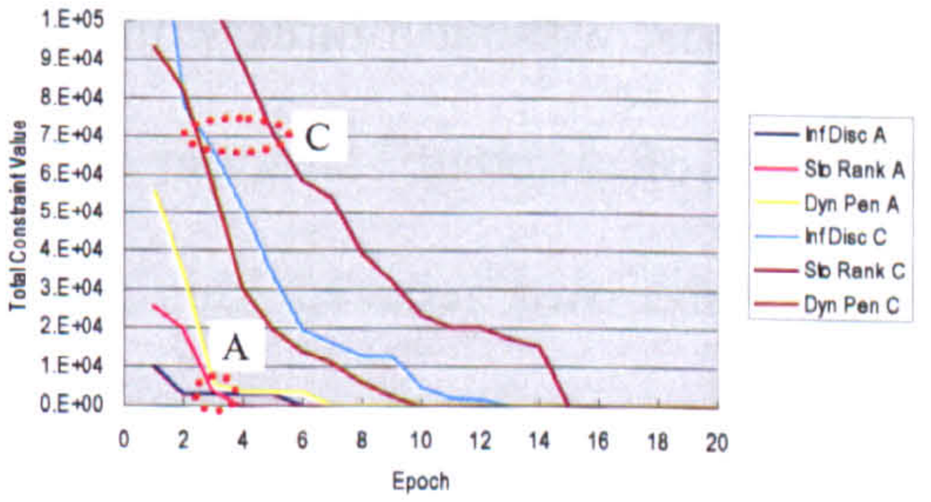
(c) Test function f_{c3}



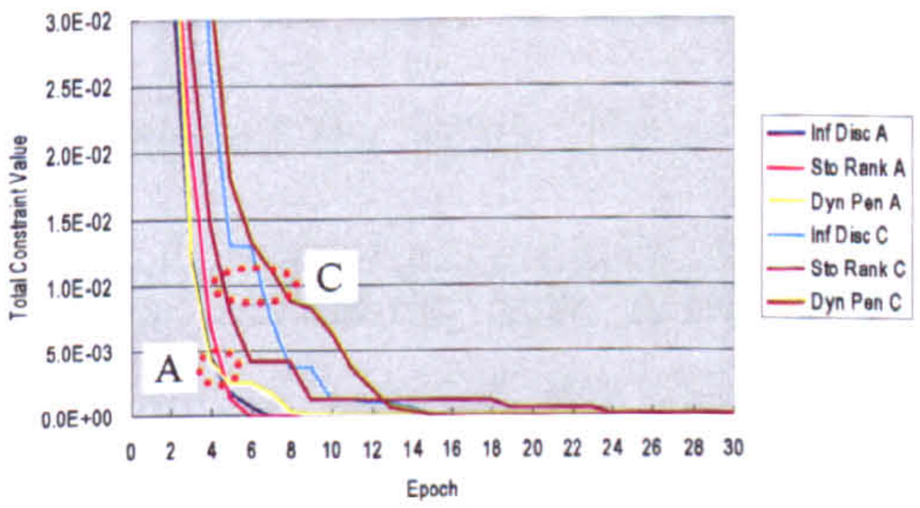
(d) Test function f_{c4}



(e) Test Function f_{c5}



(f) Test Function f_{c6}



(g) Test Function f_{c7}

Fig. 8.10. Constraint profiles of combinations A and C of EA operators at n_{pop}=5

Remarks:

a. Abbreviation:
   Inf Disc: infeasibility discrimination; Sto Rank: stochastic ranking; Dyn Pen: dynamic penalty
b. The range of the ordinate of each figure was adjusted for easy observation of the changing performance of the corresponding test function.

8.3.3 Conclusion

Since the REA was developed in the paradigms of evolutionary programming and evolution strategy, and the constraint handling operators were rank-based, it was found that the degree of contribution of the constraint handling technique to the global search was rather different to that reported in the literature of the GA (Wright 1996, Deb 2000, Coello 2002). In this work, the contribution of the constraint handling operator to the search was limited to the starting stage and feasible elite individual could be identified quickly. Then the continuing search would depend on the core EA operators of mutation, selection and recombination of REA. The constraint handling operator would still function but it was mainly responsible for identifying the feasible individuals from the infeasible ones, without a direct contribution to the elite search. This resulted in the performance of different constraint handling operators to be similar. Even when the population size was reduced, this proposition remained intact. This reinforced the primacy of the EA operators of Cauchy deterministic mutation, tournament selection and arithmetic recombination.

8.4 Summary

In this chapter, qualitative analysis of the REA was conducted, and its effectiveness has been demonstrated in a number of aspects. The reliability of optimal search of REA was studied if the evaluation function calls were limited, and the epoch might not be long enough. By using the default population of ten, the search of REA would approach within ±10% of the global optimum at only half of the epoch for a variety of constrained and unconstrained test functions. The REA could also maintain the search momentum to the end of epoch. Another study of the effectiveness of REA was

through benchmarking with the documented optimization method MGA, which features a very small population and can handle a complex simulation model with lengthy evaluation function calls. From the series of test functions, the REA with population of five (instead of default ten) could provide satisfactory results in all cases, but the MGA was not able to determine the solutions for some of the test functions. Although MGA was reported to handle different optimization problems in the literature, its effectiveness was found poor for some sorts of problems. Therefore, the REA was found to be more reliable in determining optimal solutions for a variety of problems.

The effectiveness of core EA operators of the REA was further explored from the comparative study of different constraint handling techniques. In the paradigm of GA, appropriate constraint handling operator would help the optimal search without sacrificing the potential of infeasible individuals. In the comparative studies with the published results of the TS-R method and non-dominance method, the REA could provide better or comparable performances for the related test problems. However the effectiveness of the REA did not originate from the constraint handling operator chosen, since different kinds of constraint handling operators had similar performances. This also showed the effect due to difference in the optimization paradigm of evolution strategy and GA, since mutation is emphasized in the former while crossover in the latter. This in turn affected the importance of constraint handling technique in different paradigms. For the REA, the constraint handling operator would still function, but it was mainly responsible for identifying the feasible individuals from the infeasible ones, without a significant contribution to the elite search.

As such, the constituents of the REA – Cauchy deterministic mutation, tournament selection and arithmetic recombination – could provide a reliable and

efficient optimal search for HVAC optimization problems. In next chapter, the applications of the REA in the HVAC field are described and its effectiveness demonstrated.

CHAPTER 9 APPLICATIONS OF REA IN HVAC OPTIMIZATION PROBLEMS

After confirming the effectiveness of the REA in handling a variety of constrained and unconstrained test problems with both linear and nonlinear nature, this chapter presents the performance of the REA in different HVAC optimization problems. Some of the HVAC problems were based on the developed plant simulation models, while the other problems were taken from the referred literatures in which the published results could be used for benchmarking purposes. The HVAC optimization problems were concerned with system design and energy management, and they relate strongly to engineering practice in Hong Kong. By using the REA, the performance of effective exploitation and efficient evaluation function calls was studied for these optimization problems. Topographical analysis was included where appropriate, in order to have better understanding about the relationship between the nature of the problem and the performance of the REA.

9.1 HVAC Optimization Problems under Study

9.1.1 Unconstrained optimization problems

In the author's published works (Fong *et al.* 2005a, 2006), two HVAC optimization problems have been handled with an "original EA" before the establishment of the EA Suite and formulation of the proposed REA. The "original EA" was an initial prototype EA, developed from the paradigm of evolutionary programming, with the EA operators of Gaussian deterministic mutation and proportional selection, no recombination. The two HVAC problems were related to the optimized system design and energy management of the Hong Kong local projects as follows:

a. System design of solar water heating system, through maximization of year-round

energy saving for a centralized solar water heating system for a high-rise residential building in Hong Kong (Fong *et al.* 2005a); and

b. Energy management of a subway HVAC problem, through minimization of the year-round energy consumption with the optimized monthly reset scheme of chilled water supply temperature and supply air temperature for a centralized HVAC system serving a local subway station (Fong *et al.* 2005b, 2006).

These two HVAC plant simulation models were developed with TRNSYS. They were categorized to be unconstrained, since they incorporated the related interaction constraint functions within the simulation models, so there were no extrinsic constraint functions of these optimization problems.

9.1.2 Constrained optimization problems

The applicability of the REA should also cover constrained HVAC optimization problems which would be tackled with an appropriate constraint handling technique. In Section 7.5, the proposed constraint handling operator was infeasibility discrimination (CH=1), which has stable performance with the characteristics of not deforming the fitness landscape throughout the search progress, and the degree of infeasibility can be used to indicate the potential of infeasible individuals in global search. In order to cover the application of the REA on the HVAC problems with extrinsic constraint functions, two more optimization problems were used:

a. Design of duct system; and

b. Energy management of HVAC heat rejection system.

These two optimization problems were included since they have the practical value in HVAC applications. The problem of "design of duct system" can be applied for typical

duct sizing by considering the spatial constraints, pressure balancing requirement and velocity limitations. The problem of "energy management of HVAC heat rejection system" can demonstrate the composition of empirical mathematical plant models and different kinds of interaction constraints. In addition, there are referred literatures with the published information about the problem formulation and results. The former problem was based on the ASHRAE Handbook Fundamentals (ASHRAE 2005) and the work of Asiedu *et al.* (2000), while the latter was developed with reference to the work of Lu *et al.* (2004). In Asiedu's and Lu's publications, the GA with penalty-based method was adopted to handle these two constrained HVAC optimization problems. In order to implement the REA, both the objective and constraint functions were developed according to the system nature, as well as with suitable reference to those two literatures, so benchmarking of optimization results could be carried out at last.

9.1.3 Summary of HVAC optimization problems

To investigate the possible applications of the REA and the simulation-optimization EA Suite in the HVAC field in Hong Kong, the problems were grouped into the unconstrained and constrained aspects, either design or energy management optimization problems:

a. Unconstrained design optimization problem – solar water heating system
b. Unconstrained energy management optimization problem – subway HVAC system
c. Constrained design optimization problem – duct system
d. Constrained energy management optimization problem – heat rejection system

The development of the related HVAC system and the problem formulation on the optimization platform of the EA Suite, is discussed in the ensuing sections.

9.2 Implementation of REA in HVAC Optimization Problems

9.2.1 Unconstrained design optimization problem – solar water heating system

9.2.1.1 Design of solar water heating system for high-rise residential building

The design of a proposed centralized solar water heating system for a high-rise residential building in Hong Kong is shown in Fig. 9.1 (Fong *et al.* 2005a, Chow *et al.* 2006). The whole plant simulation model was developed with TRNSYS 15.3, which adopted the DFP algorithm (as mentioned in Section 2.1.2) for solving a set of non-linear equations from the simulation components, such as the thermal solar collector and hot water calorifier (SEL 2000).

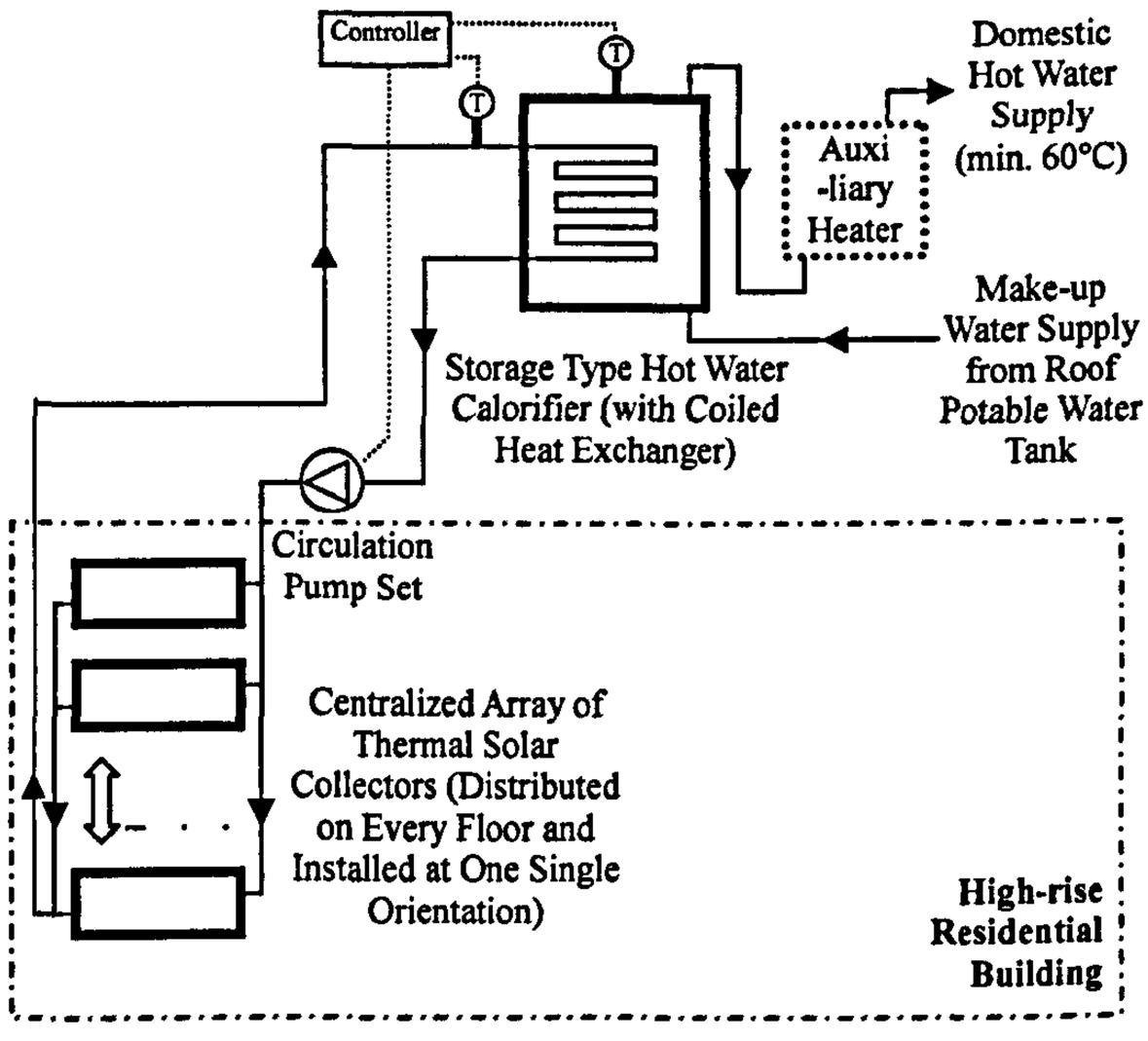


Fig. 9.1. Typical schematic diagram of the simulated solar water heating system for a high-rise residential building

The flat plate collector was selected rather than evacuated tubes since it is more effective in Hong Kong with relatively warm winters and significant diffuse solar radiation around the year (SPARE 2002). In this study, it was assumed that the array of all the solar collectors was installed to one single orientation of the building. The

residential building under study had 28 stories, a floor to floor height of 3 m, and eight apartments per floor. A total of 840 m^2 (or 3.75 m^2 per apartment) of the collectors was applied. The collector fin efficiency factor, emittance and absorptance of the absorber plate were 0.78, 0.03 and 0.947 respectively. In this simulation, the global horizontal radiation, direct normal radiation, air dry bulb temperature and wind speed were all based on the weather data of the newly developed typical meteorological year for Hong Kong (Chan *et al.* 2006). The hourly variation profile of solar radiation is shown in Fig. 9.2. The global horizontal and direct normal radiation were input to the simulation component of solar radiation processor, which adopted the Reindl model (Reindl *et al.* 1990) to resolve the beam and diffuse incident solar radiation on the solar collectors with different surface azimuths and tilt angles. The hourly dry-bulb temperature and wind speed were linked to the component models of thermal solar collector, pipe and hot water calorifier, so that the convective and radiative heat losses from such equipment could be evaluated with changing weather conditions.

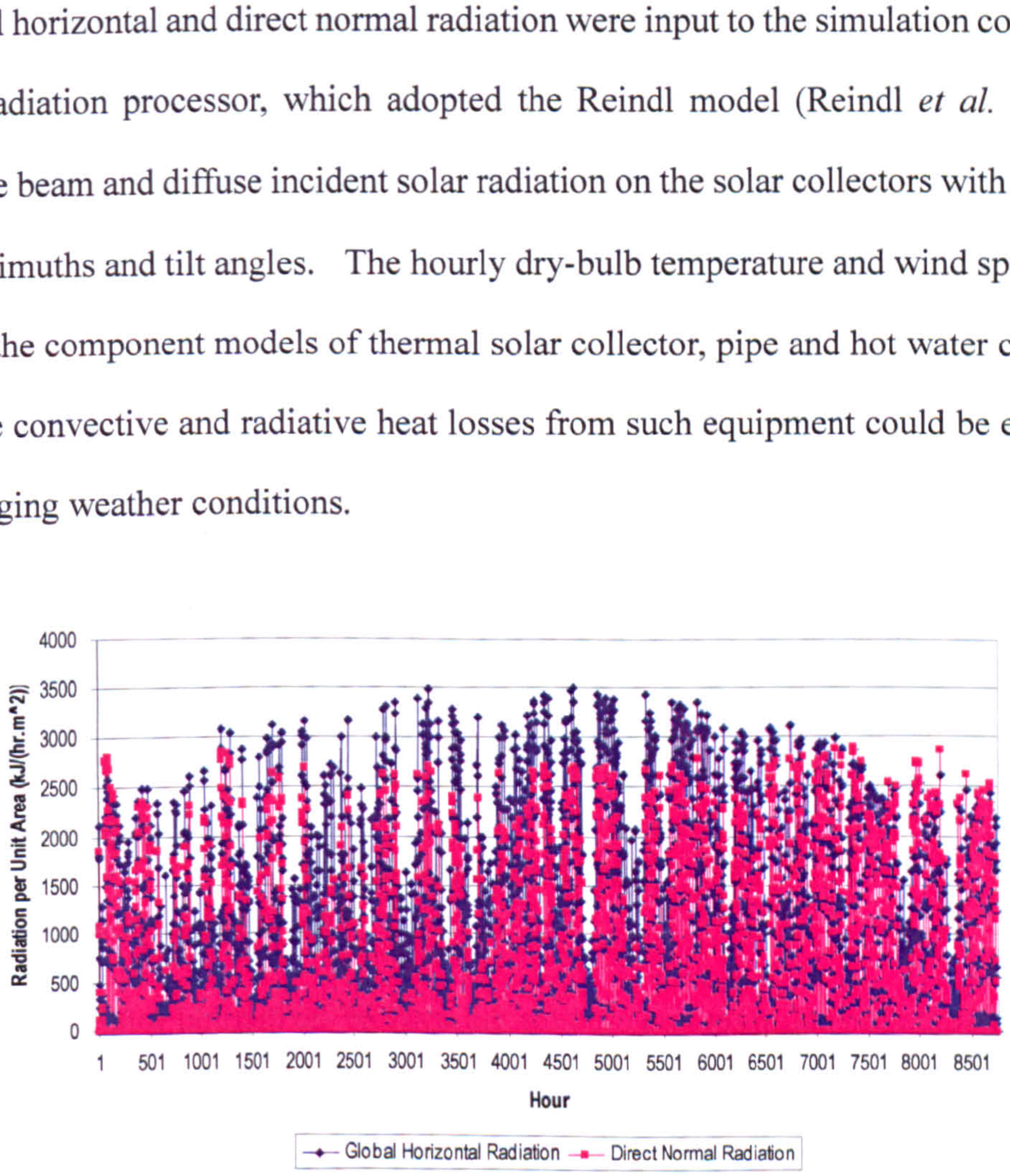


Fig. 9.2. Hourly variation profile of solar radiation in Hong Kong throughout a year

The thermal gain from the solar collectors was stored in a hot water calorifier

which had thermal stratification with 5 isothermal tank nodes. The hot water calorifier had a copper coiled heat exchanger to provide indirect heating and prevent any cross contamination between the circulating water through the solar collectors and the make-up water from the potable water tank. The storage capacity of the hot water calorifier was 36 m^3, which was corresponding to the domestic hot water (DHW) daily consumption of 160 litres per apartment and the total number of 224 apartments in the high-rise residential building. The daily consumption profile of DHW, as shown in Fig. 9.3, was associated to the water supply to and from the hot water calorifier model. The profile was based on local design practice, with a moderate demand in the morning, a peak demand at the night and a minimum demand at the rest of the period. The make-up water temperature varied with the month and ranged from 15°C to 25°C.

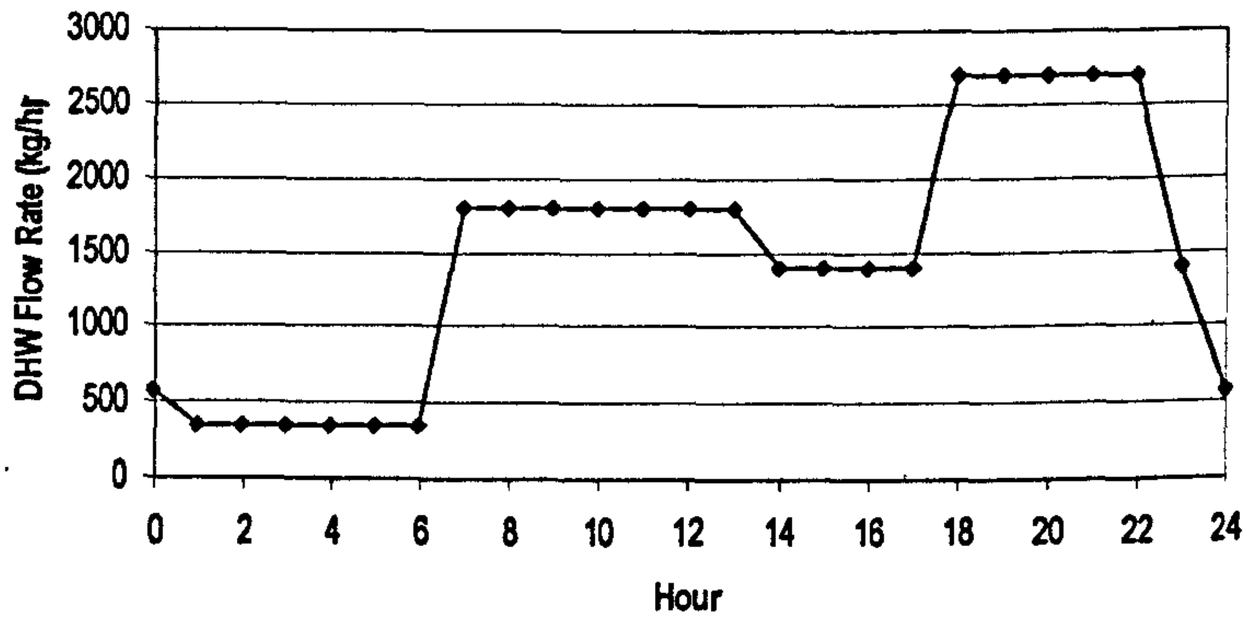


Fig. 9.3. Daily DHW consumption profile

A circulation pump set was used for the array of the thermal solar collectors. The on-off operation of the pump set was primarily according to the operating schedule from 06:00 to 19:00 daily, but could be overridden by the temperature differential signal between the manifold outlet of the solar collectors and the calorifier as shown in Fig. 9.1. Once the temperature of the manifold outlet was lower than that of the calorifier, the

pump set would be turned off. In order to prevent a hunting effect in pump operation, 8°C and 2°C of higher and lower dead bands were set respectively. DHW was drawn directly from the hot water calorifier, and an in-line auxiliary heater was operated whenever the DHW supply temperature was below 60°C, in order to comply with the local code for the prevention of Legionnaires' disease (CPPLD 2000). The rated pump power was determined by considering the flow rate and the total pressure drop of the entire closed-loop circuitry according to the local design practice, then the pump flow rate and pump power were 47000 kg/hr and 7400 kJ/hr respectively.

9.2.1.2 Problem variables

Since the optimization was targeted on the overall performance of the centralized solar water heating system, the design variables were identified from different areas of the system design. In order to study the potential energy saving due to the application of the proposed solar water heating system, the optimal values of those design parameters should return a maximum yearly energy saving of the entire system as compared to the conventional domestic electric heating. After careful consideration from the system design in Fig. 9.1, the following design variables would affect the optimal performance of the solar water heating system:

- tilt angle of solar collectors β;
- surface azimuth of solar collectors γ_s;
- storage capacity of hot water calorifier V_{cal}; and
- mass flow rate of circulation pump set m_{pp}.

It was necessary to determine the best β and γ_s of the thermal solar collectors in order to obtain the maximum amount of thermal gain through both the incident beam and diffuse

solar radiations throughout a year. For V_{cal} and m_{pp}, although preliminary design values (36 m^3 and 47000 kg/hr respectively) were determined from the design practice, it was necessary to analyze carefully for any compromising effect on retaining useful thermal energy in respect of the daily DHW consumption profile. Therefore β, γ_s, V_{cal} and m_{pp} became the problem variables to achieve the maximum energy saving of the solar water heating system against electric heating for the high-rise residential building.

9.2.1.3 Objective function

The objective function of this optimization problem was to maximize the year-round (totally 8760 hours) saving in electricity consumption E_{saving} by using the solar heating E_{sheat}, against that by the conventional electric heating E_{eheat} as follows:

$$\text{maximize } E_{saving} = \sum_{i=1}^{8760} (E_{eheat,i} - E_{sheat,i}) \tag{9.1}$$

E_{saving} became the function value used to prioritize the fitness of different combinations of the problem variables.

For electric heating alone, energy consumption was mainly determined by the conventional domestic electric water heaters. On the other hand for solar heating, the total energy consumption would include the energy of the circulation pump set for the array of solar collectors E_{pp} and that of the centralized auxiliary electric heater E_{aheat}. Whenever the DHW supply temperature was lower than $T_{dhw,min}$ of 60°C (i.e. the minimum DHW supply temperature stipulated by the local regulation), the auxiliary electric heater would be deployed. Therefore for each simulation hour, the relationship for E_{eheat} and that for E_{sheat} would be as follows:

$$E_{eheat} = m_{dhw} C_{pw} (T_{dhw,min} - T_{mw}) \tag{9.2}$$

$$E_{sheat} = E_{pp} + E_{aheat} \tag{9.3}$$

$$E_{aheat} = \max\{[m_{dhw}\, C_{pw}\, (T_{dhw,min} - T_{hw,o})], 0\} \tag{9.4}$$

where,

m_{dhw}: mass flow rate of domestic hot water (kg/hr)

$T_{hw,o}$: hot water outlet temperature of calorifier (°C)

T_{mw}: make-up potable water temperature (°C)

In Eq (9.4), the "max" function will return a value for E_{aheat} only if the first quantity is greater than zero. Since there were occasions when $T_{hw,o}$ was greater than $T_{dhw,min}$ (in case of receiving abundant incident solar radiation), then the first quantity would become a negative value, hence E_{aheat} should be zero. This implied that the auxiliary heating was not necessary and no energy consumption took place.

In the plant simulation model, m_{dhw} and T_{mw} were the inputs from the profiles of DHW consumption and make-up water temperature. With the constant nature of C_{pw} (4.19 kJ/(kg·K)) and $T_{dhw,min}$ (60°C), E_{eheat} could be determined for each hour by Eq (9.2). In Eq (9.3), E_{sheat} could be determined by summing up E_{pp} and E_{aheat}, where the former was assumed to be directly proportional to the optimized m_{pp}. This was because the pump pressure would be relatively constant, for the pipe diameter of the circulation circuit would be sized according to a recommended range of pressure drop per unit length under good design practice for pipe sizing. On the other hand the calculation of E_{aheat} was not so straightforward. In Eq (9.4), $T_{hw,o}$ would be a function of the four problem variables β, γ_s, V_{cal} and m_{pp} within the plant simulation model. Therefore, $T_{hw,o}$ became the important output from the plant simulation model in order to determine E_{aheat}, hence E_{sheat} and eventually E_{saving} in Eq (9.1).

9.2.1.4 Parametric studies of problem variables

In order to preview how the problem variables could affect the year-round energy saving, it was decided to prepare suitable parametric plots to study the fitness landscape. In this problem, since there were four variables, it was a four-dimensional optimization problem and therefore not possible to present graphically the fitness value E_{saving} against all the problem variables. Therefore two parametric plots were prepared, one for E_{saving} against β and γ_s is shown in Fig. 9.4, while another for E_{saving} against m_{pp} and V_{cal} is shown in Fig. 9.5. From Fig. 9.4, maximum E_{saving} might be achieved when β was between 0 to 45 degree and γ_s was between -45 to 45 degree south. On the other hand from Fig. 9.5, maximum E_{saving} might be found when m_{pp} was between 20000 to 25000 kg/hr, while V_{cal} was between 35 to 40 m^3.

However, the parametric plots could only provide a snapshot of the preliminary relationship of the fitness value to any two of the problem variables. If these four parameters were optimized simultaneously, the fitness values might be different. This would depend on the interrelationship of the problem variables in the plant simulation model. Nevertheless, these parametric plots would furnish a preliminary understanding for the development of the details of the ongoing optimization.

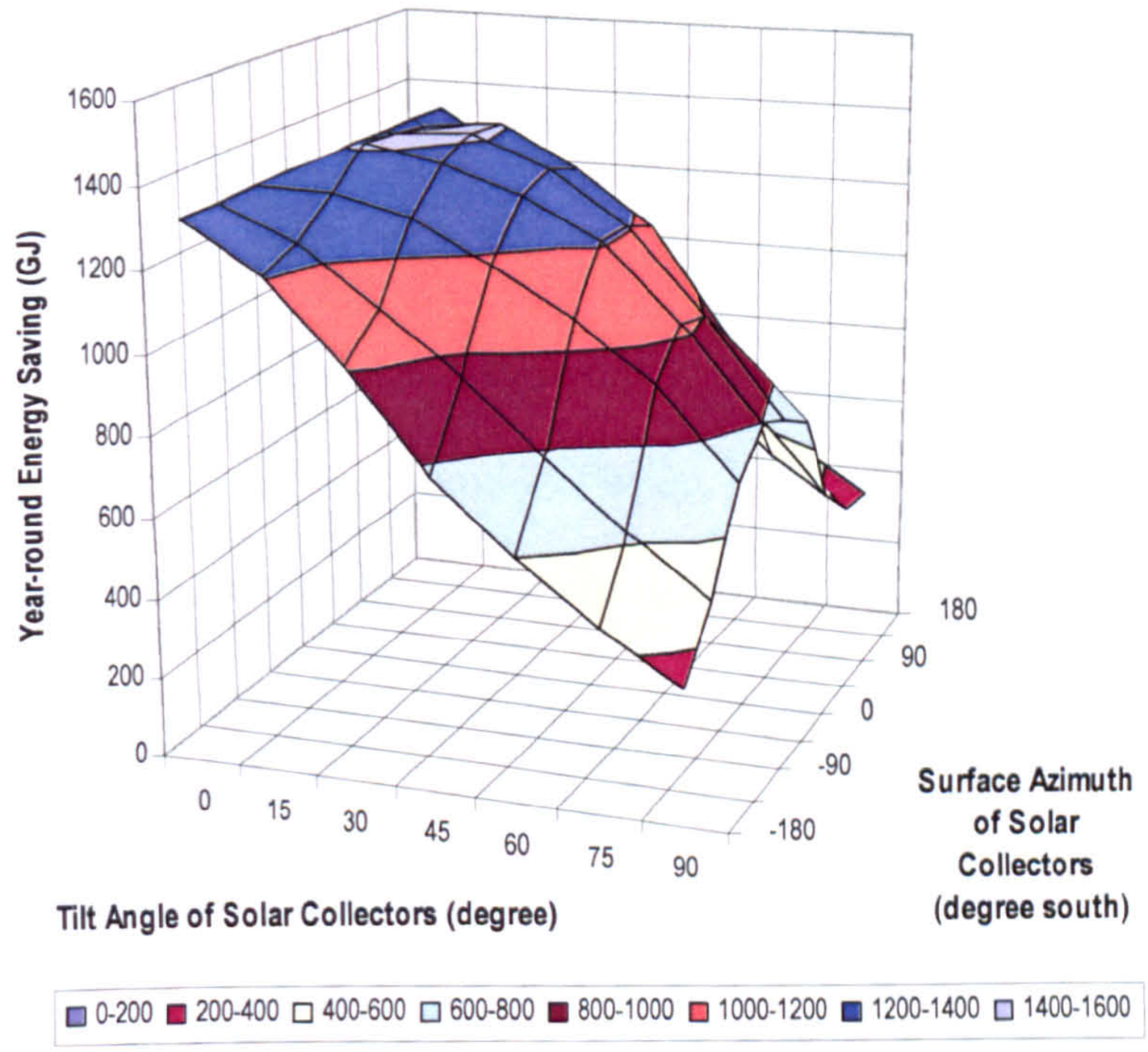


Fig. 9.4. Parametric surface of year-round energy saving vs. tilt angle and surface azimuth of solar collectors (for $V_{cal} = 36\ m^3$ & $m_{pp} = 47000$ kg/hr)

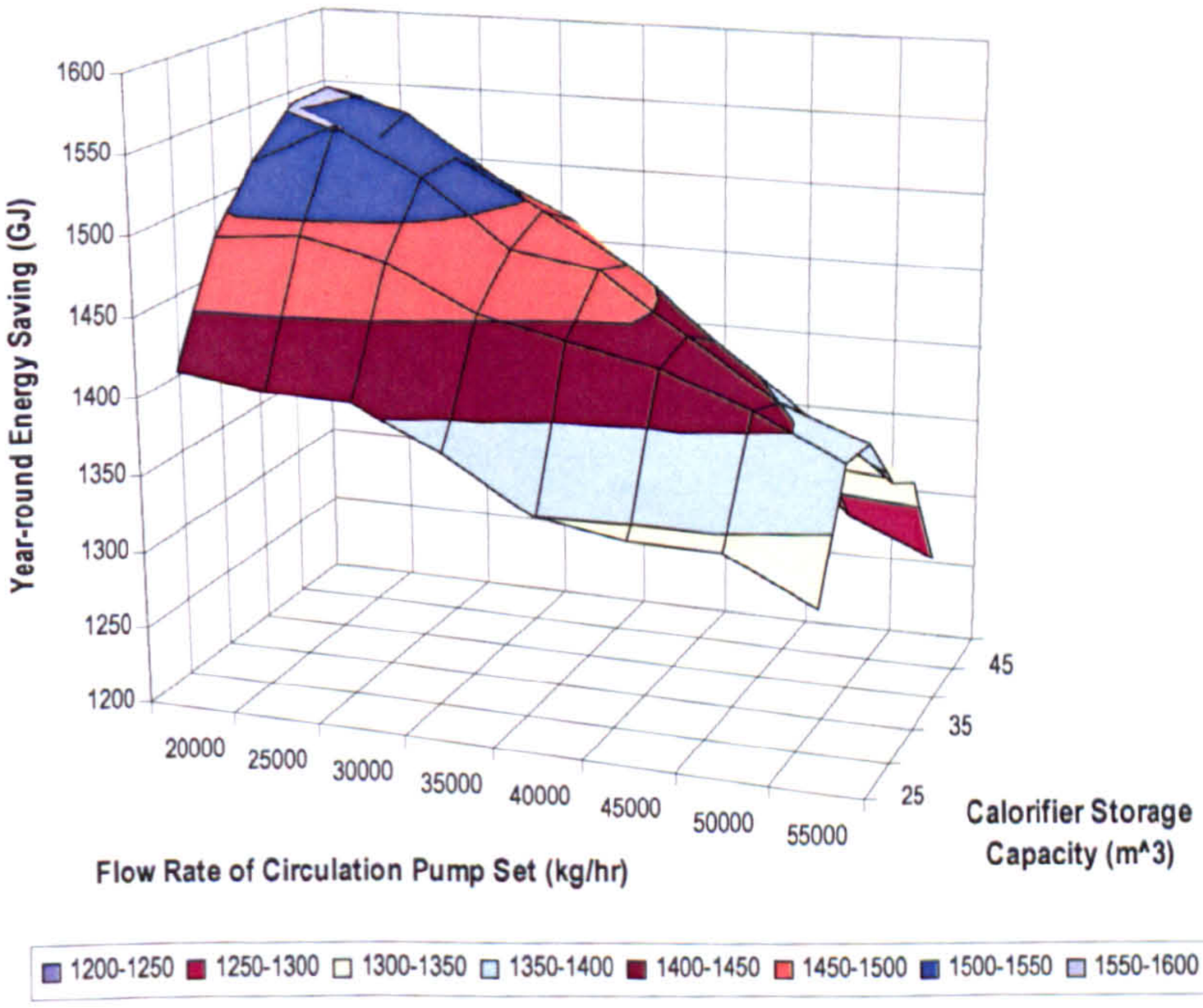


Fig. 9.5. Parametric surface of year-round energy saving vs. pump flow rate and calorifier storage capacity (for $\beta = 22.3°$ & $\gamma_s = 0°$ due south)

9.2.1.5 Preview of fitness topography

For the unconstrained solar water heating design problem, the topography of the fitness landscape was illustrated through 3D plots with narrower ranges for the problem variables around the global optimum, as illustrated in Figs. 9.7 and 9.8. The possible region of global optimum is labeled in the yellow box, the local optimum/optima in white. The multimodal nature of the fitness landscapes is demonstrated. Again, these two topographical previews were not the actual landscapes, since only two problem variables were involved in each diagram, not all the four of the solar water heating optimization problem.

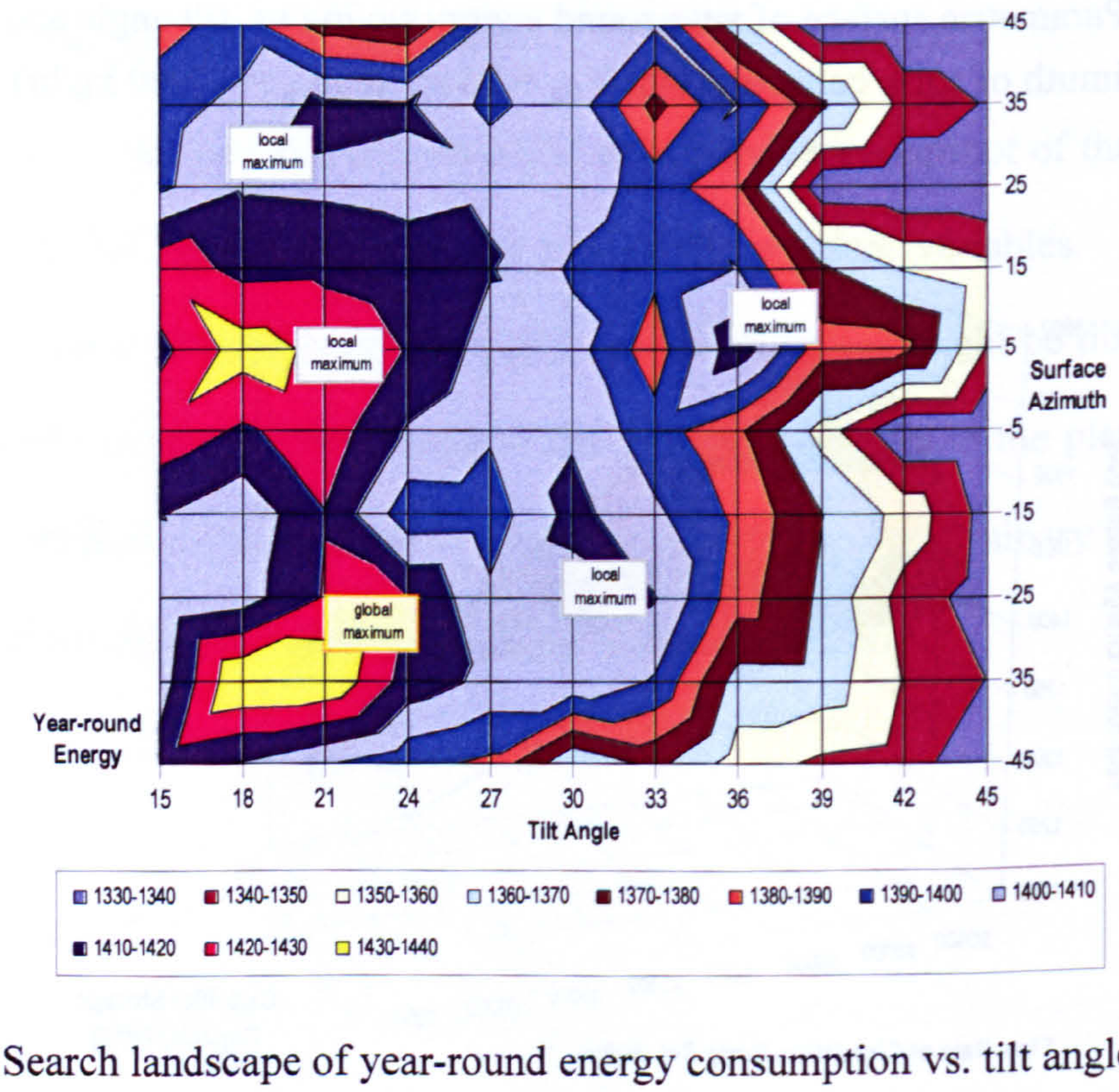


Fig. 9.7. Search landscape of year-round energy consumption vs. tilt angle and surface azimuth of solar water heating design problem (for $V_{cal} = 36\ m^3$ & $m_{pp} = 47000$ kg/hr)

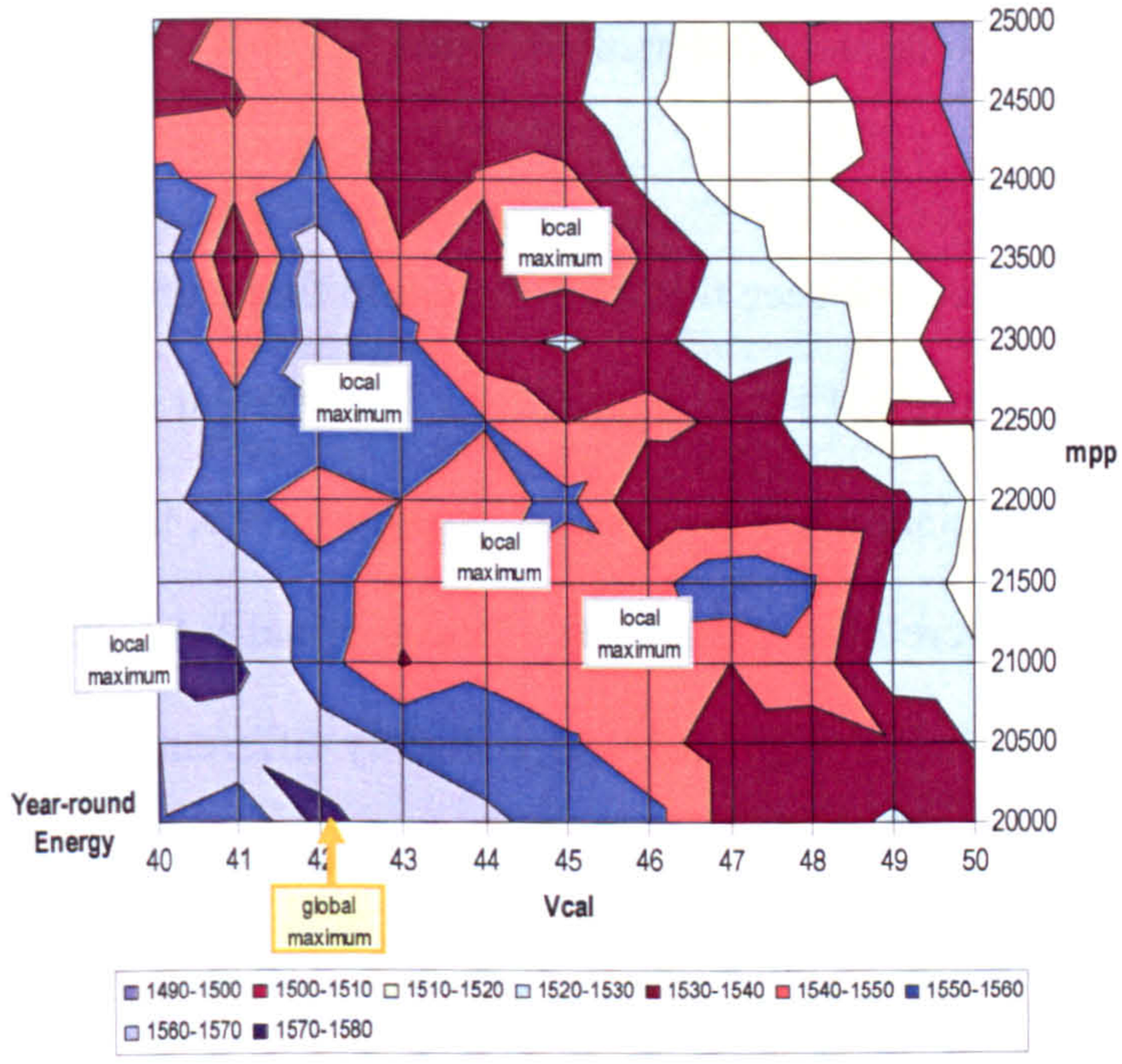


Fig. 9.8. Search landscape of year-round energy consumption vs. storage capacity and pump flow rate of solar water heating design problem (for $\beta = 22.3°$ & $\gamma_s = 0°$ due south)

9.2.1.6 Implementation and results

Through maximization of the annual energy saving of the system, the optimal results of the design parameters, including the tilt angle and surface azimuth of the thermal solar collectors, the capacity of the stratified hot water calorifier, and the flow rate of the centralized circulation pump for the distributed solar collectors, could be determined for design purposes. The old version of EA with Gaussian deterministic mutation and proportional selection (no recombination) [MU=1 + SE=3] was originally used for this optimization problem (Fong *et al.* 2005a). The efficiency of the optimization method was crucial to this problem, since the running time at n_{pop} of 10 and $epoch_{max}$ of 50 was about 34 hours using a personal computer (configuration of Pentium 4 and 1 GB RAM). A re-run was carried out with the REA (arithmetic recombination + Cauchy deterministic mutation + tournament selection) for both $epoch_{max}$ of 50 and 30.

All the optimization results are presented in Table 9.1 and Fig. 9.6 for comparison purposes.

From Table 9.1 it can be seen that the REA could find a greater year-round energy saving compared to the original EA at n_{pop} of 50. The REA at n_{pop} of 30 gave a better result than that of the original EA at n_{pop} of 50. From Fig. 9.6, the convergence rates of both cases of the REA were higher than that of the original EA. It was found that the optimal design values from the original EA were slightly different from those of the REA. Therefore it is important to make sure that the optimization method can search for the optimal design variables effectively within the prescribed span of epoch, otherwise there could be a discrepancy in the design outcome, mainly the surface azimuth of the solar collectors in this case. As a result, the REA could achieve a better optimal search in this solar water heating design problem.

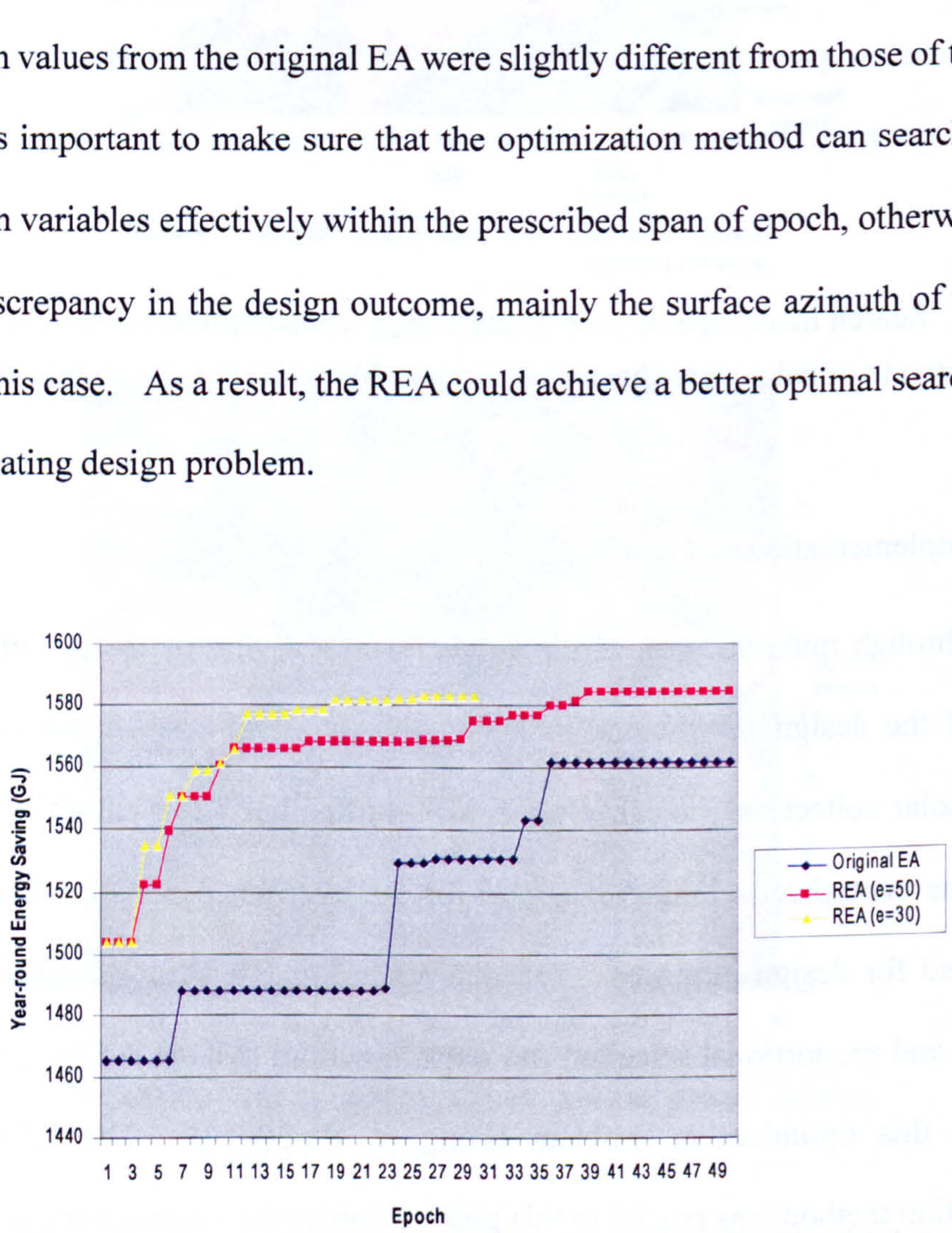


Fig. 9.6. Search profiles of original EA and REA for solar water heating design problem

Table 9.1. Optimization results of original EA and REA for solar water heating design problem

	Original EA (Fong *et al.* 2005a)	REA at $epoch_{max}$ = 50	REA at $epoch_{max}$ = 30
Implementation			
EA operators	(No recombination) Gaussian det. mutation Proportional selection	Arithmetic recomb. Cauchy det. mutation Tournament selection	Arithmetic recomb. Cauchy det. mutation Tournament selection
Population size	10	10	10
Epoch of termination	50	50	30
Computational time (hr)	34	34	20
Results			
Year-round energy saving (GJ)	1561.9	1584.8 (1.5% ↑)	1583.0 (1.4% ↑)
Optimal tilt angle (degree)	20.6	23.4	22.3
Optimal surface azimuth (degree due south)	15.5	6.2	7.7
Optimal storage capacity (m^3)	41.6	39.9	40.4
Optimal flow rate of circulation pump (kg/hr)	20426.9	22566.0	22501.7

9.2.2 Unconstrained energy management optimization problem – subway HVAC system

9.2.2.1 HVAC system for local subway station

This was an optimization problem concerning the energy management of a centralized HVAC system for a subway station in Hong Kong (Fong *et al.* 2005b, 2006) as shown in Fig. 9.9. A component-based model of the entire HVAC system was built up with TRNSYS, together with the required control and operating functions in order to reflect the year-round operation of the actual equipment within the air side and water side systems. This plant simulation model would be expected to have a suitable reset scheme

of the supply air temperature for the AHUs inside the platform and concourse, as well as that of the chilled water supply temperature for the chiller plant of the local subway station.

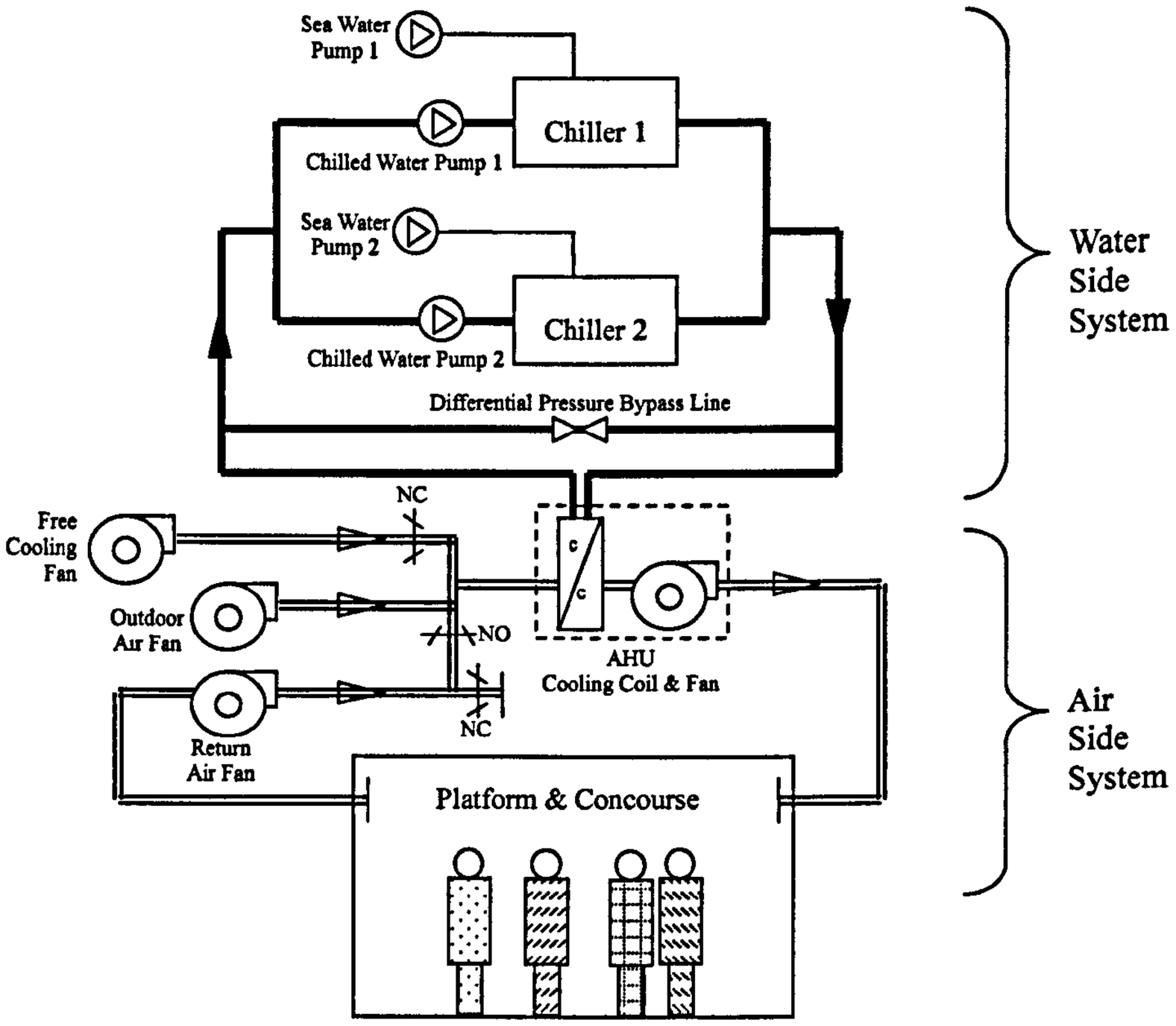


Fig. 9.9. Schematic diagram of HVAC system for subway station

From Fig. 9.9, there were two chillers, each with cooling capacity of 1820 kW (518 TR), together with the associated chilled water and sea water pumps. Differential pressure bypass was applied as a hydronic decoupling during the part load operation. For the air side system, there were a number of AHUs and fans in the actual installations. In order to simplify the model and hence save the simulation time, particularly later in the optimization process, it was assumed that the same kind of air side equipment ran in the same operating mode together. Therefore the air side was lumped into one AHU, one outdoor air fan, one return air fan, and one free cooling fan. The power ratings of the different equipment are summarized in Table 9.2.

Table 9.2. Equipment power rating of the subway HVAC system

Item	Equipment	Power Rating (kW)	Relative Percentage (%)	Sub-total Percentage
1	Water-cooled chiller 01	348.66	26.01	
2	Water-cooled chiller 02	348.66	26.01	Total 59.78% for the
3	Chilled water pump 01	30	2.24	water side system
4	Chilled water pump 02	30	2.24	(52.02% for chillers)
5	Sea water pump 01	22	1.64	
6	Sea water pump 02	22	1.64	
7	AHU fan (lumped)	222	16.56	
8	Outdoor air fan (lumped)	41.55	3.10	Total 40.22% for the
9	Return air fan (lumped)	163	12.16	air side system
10	Free cooling fan (lumped)	112.5	8.39	
	Total	1340.37	100	

Local practice for a typical commercial building project would give the percentage allocation of power between the water side system and air side system as usually about 80% to 20% for a similar configuration. However from Table 9.2, the percentage allocation was about 60% to 40%, therefore it would pay more attention to the operational and control aspect of the air side system.

To evaluate the opportunities for energy management of the HVAC system attention must be focused on the installed features of the equipment. The chilled water pumps and AHU fans were not of the variable speed type. Since constant speed chilled water pumps and differential pressure bypass were installed, reset of the chilled water supply temperature $T_{chw,sp}$ might be more beneficial for this type of equipment. Similarly, as two-speed supply air fans were installed in the air side system, reset of the supply air temperature $T_{sa,sp}$ might also be useful to enhance energy efficiency. The saving potential would depend on the complexity and configuration of the HVAC system, and there being no clear guidelines as to the optimal combination of $T_{chw,sp}$ and $T_{sa,sp}$ for the operation, a suitable optimization technique is required for the study of this problem.

9.2.2.2 Problem variables and objective function

The problem variables to be optimized included:

- set point of chilled water supply temperature of chiller $T_{chw,sp}$; and
- set point of supply air temperature of AHU $T_{sa,sp}$.

This study of application of the REA, was mainly focused on minimization of monthly energy consumption with both the optimal $T_{chw,sp}$ and $T_{sa,sp}$. Each individual contained both $T_{chw,sp}$ and $T_{sa,sp}$, and the fitness function was based on monthly evaluation, as shown in Eqs (9.5) and (9.6). For each operating hour i (i = 1, 2, …, k), the hourly total energy consumption $E_{total,i}$ was determined for all the operating equipment in response to the cooling load and weather condition in that hour. Then the monthly energy consumption was determined for the corresponding operating hours within the month.

$$E_{total} = \sum_{i=1}^{k} E_{total,i} \qquad (9.5)$$

and, $$E_{total,i} = E_{WCC,i} + E_{CHP,i} + E_{SWP,i} + E_{AHU,i} + E_{OAF,i} + E_{RAF,i} + E_{FCF,i} \qquad (9.6)$$

where,

k { = 598 hours (31 × 19) for Jan, Mar, May, Jul, Aug, Oct and Dec;
  = 570 hours (30 × 19) for Apr, Jun, Sep and Nov; or
  = 532 hours (28 × 19) for Feb only.
  (based on 19-hour daily operation of the HVAC plant)

$E_{WCC,i}$: hourly energy consumption of water-cooled chillers (kJ)

$E_{CHP,i}$: hourly energy consumption of chilled water pumps (kJ)

$E_{SWP,i}$: hourly energy consumption of sea water pumps (kJ)

$E_{AHU,i}$: hourly energy consumption of AHU fan (kJ)

$E_{OAF,i}$: hourly energy consumption of outdoor air fan (kJ)

$E_{RAF,i}$: hourly energy consumption of return air fan (kJ)

$E_{FCF,i}$: hourly energy consumption of free cooling fan (kJ)

The bounds of the problem variables were as follows:

$$5\ °C \leq T_{chw,sp} \leq 8\ °C$$

$$13\ °C \leq T_{sa,sp} \leq 19\ °C$$

9.2.2.3 Preview of search landscape

Before the actual run of the subway HVAC system in the EA Suite, it would be useful to have some understanding of the fitness landscape, like the case of solar water heating design in Section 9.2.1.4. For the subway problem, there were two problem variables $T_{chw,sp}$ and $T_{sa,sp}$, so a two-dimensional search surface was developed for a parametric study within their bounds. For the preview purpose, the fitness landscape of year-round energy consumption of the entire HVAC system is plotted in Fig. 9.10. In order to have a thorough understanding, the parametric surfaces of the water side system and air side system were also developed in Figs. 9.11 and 9.12 respectively.

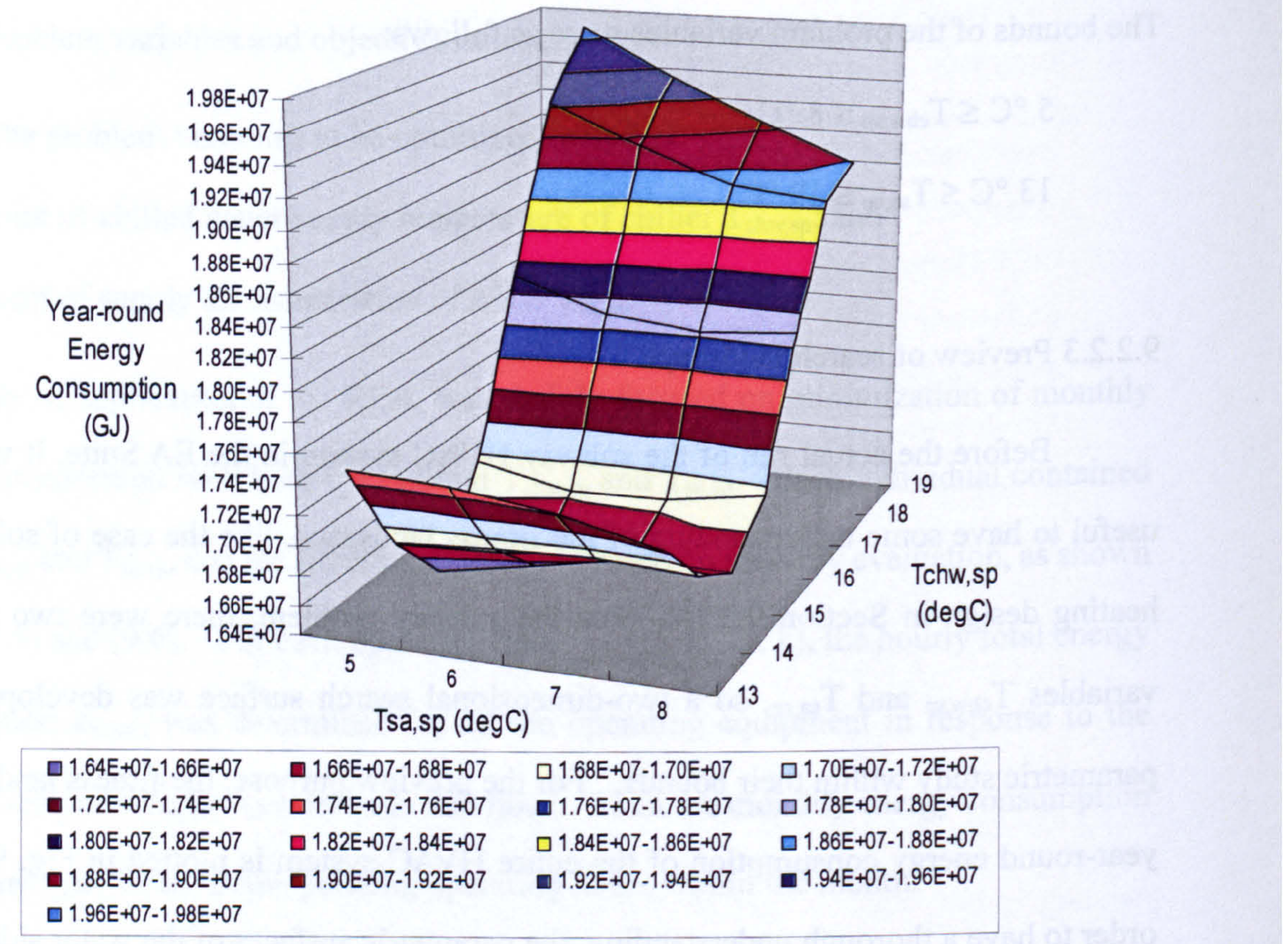


Fig. 9.10. Fitness landscape of year-round energy consumption against $T_{chw,sp}$ and $T_{sa,sp}$ for entire subway HVAC system

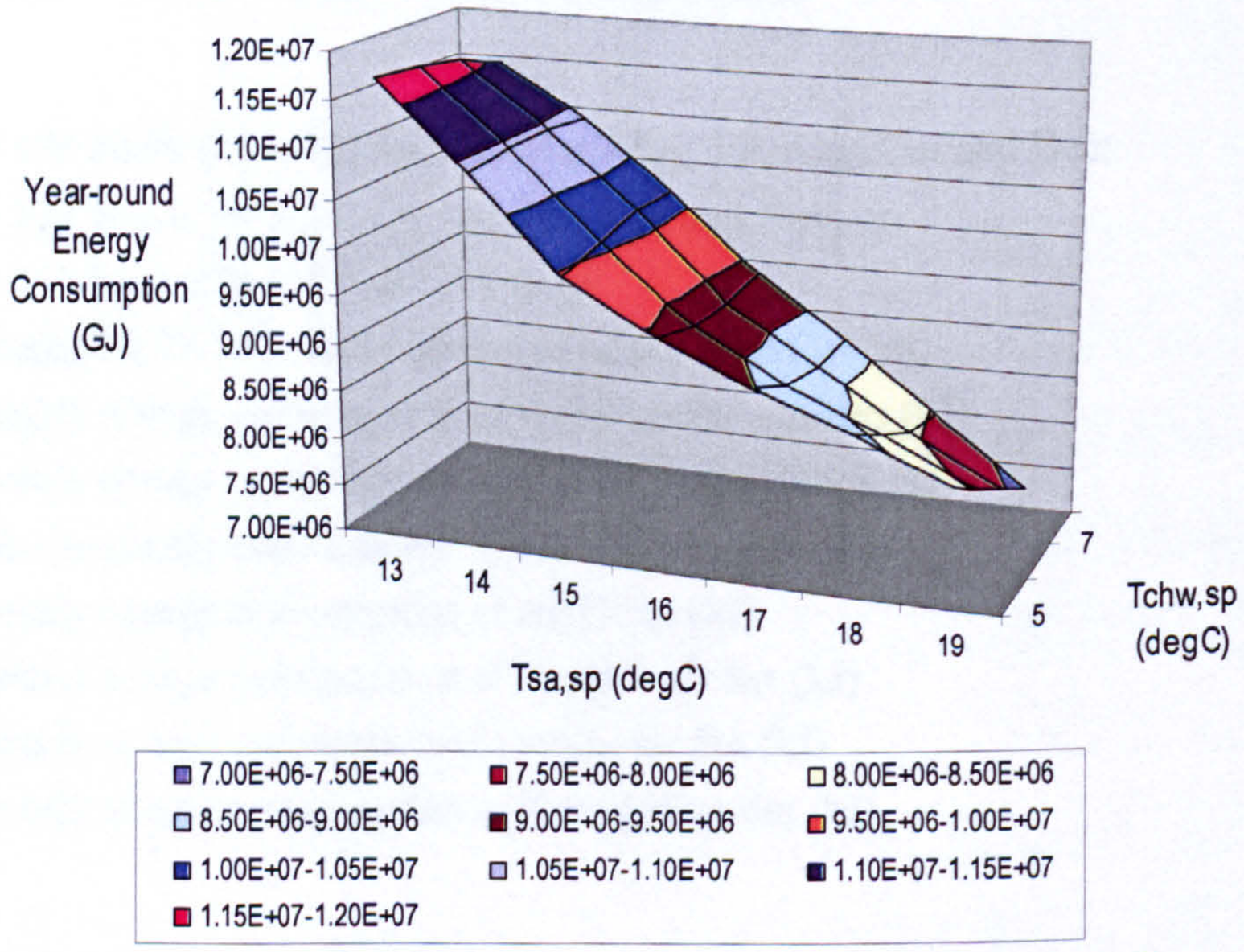


Fig. 9.11. Parametric surface of year-round energy consumption against $T_{chw,sp}$ and $T_{sa,sp}$ for water side system

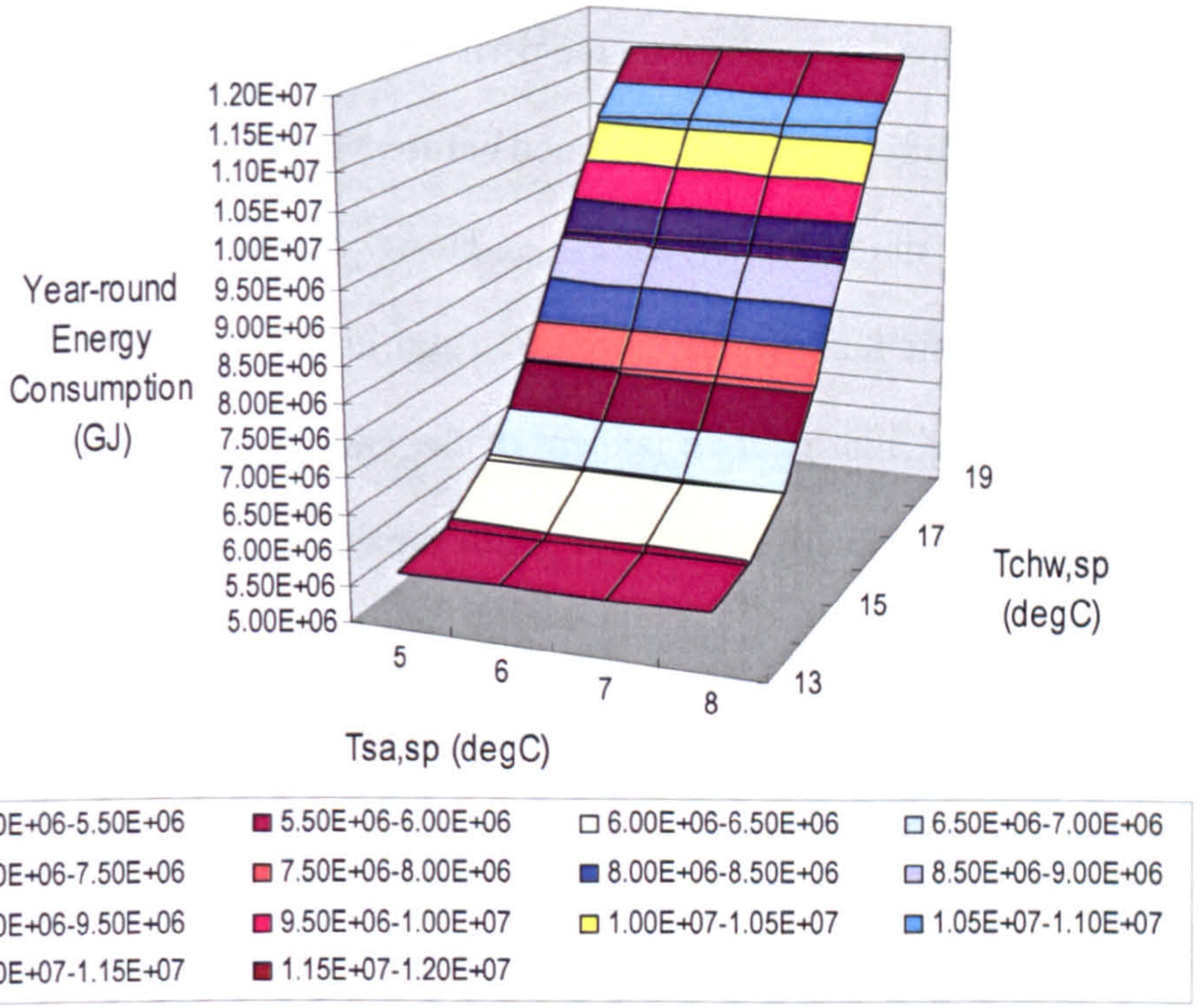


Fig. 9.12. Parametric surface of year-round energy consumption against $T_{chw,sp}$ and $T_{sa,sp}$ for air side system

From Fig. 9.10, it can be seen that the shape of the fitness landscape was like a "chair", the surface smoothness was fair and local minima or maxima were not prominent. As a minimization problem, the region with $T_{chw,sp}$ near 5 °C and $T_{sa,sp}$ near 19 °C would not be of practical concern, since there were relatively high fitness values involved. Figs. 9.11 and 9.12 gave a better understanding of the resultant landscape of the entire system found in Fig. 9.10. From Fig. 9.11, it can be seen that it was advantageous to have a higher $T_{sa,sp}$ for the water side system, for two major reasons. The first reason was that higher chilled water temperature would lead to a higher $T_{sa,sp}$, so in turn the chiller would consume less energy due to better COP. The second reason was that when $T_{sa,sp}$ was high during the mid-seasons, there would be more frequent occasions when the outdoor air enthalpy was less than the space air enthalpy. The air side would then have more opportunities to operate in free cooling mode, thus stopping operation of the chiller plant

and its associated pumps. The effect on the performance of the air side system as shown in Fig. 9.12 was very different. As mentioned before, there would be more opportunities to run in free cooling mode for higher $T_{sa,sp}$. However when the chillers were out of operation the supply air fans would more often run in full speed mode instead of half speed. Owning to the high relative percent of the energy consumption of the air side equipment as shown in Table 9.2, there should be a pay-off between the operation of the water side system and air side system with different $T_{chw,sp}$ and $T_{sa,sp}$. which would be around the "seating-area" of the chair-like fitness landscape in Fig. 9.10.

9.2.2.4 Implementation and results

With the available published results from the original EA (Fong *et al.* 2006), the subway HVAC problem was optimized again using the REA (arithmetic recombination + Cauchy deterministic mutation + tournament selection). The results were compared and shown in Table 9.3. In the table, the monthly energy consumption of all the months as determined by the REA were lower than those by the original EA. The former results were within the corresponding region around the global optimum as shown in the search contour plots of the fitness landscapes in Figs. 9.13 to 9.20, but most of the latter results did not lie within those regions, as encompassed by corrugated line type in Table 9.3. As a result, the REA found better optimal results for different topography of the fitness landscapes in different months, particularly in the case of those months with multimodal characteristics, namely April and October, as shown in Fig. 9.16 and 9.18 respectively.

Table 9.3. Performance comparison between original EA and REA for energy management problem of subway HVAC system

Month	Results of original EA with Gaussian deterministic mutation and proportional selection (no recombination) (as published in Fong *et al.* 2006)		Results of REA with arithmetic recombination, Cauchy deterministic mutation and tournament selection	
	Monthly energy consumption ($\times 10^3$ GJ)	$T_{chw,sp}$, $T_{sa,sp}$ (°C, °C)	Monthly energy consumption ($\times 10^3$ GJ)	$T_{chw,sp}$, $T_{sa,sp}$ (°C, °C)
Jan	799.47	6.7, 14.0	749.82	5.0, 13.8
Feb	751.55	6.7, 14.3	692.59	5.0, 13.9
Mar	879.27	5.0, 15.3	879.14	5.0, 15.3
Apr	1282.20	5.4, 15.5	1261.95	6.0, 15.8
May	1677.77	8.0, 13.7	1672.68	8.0, 13.0
Jun	1688.71	8.0, 13.6	1683.18	8.0, 13.0
Jul	1770.92	8.0, 13.6	1763.73	8.0, 13.0
Aug	1775.92	8.0, 13.6	1769.74	8.0, 13.0
Sep	1685.58	8.0, 13.6	1680.75	8.0, 13.0
Oct	1625.73	8.0, 14.2	1625.32	8.0, 13.1
Nov	1125.92	5.0, 15.3	1124.81	5.0, 15.2
Dec	649.35	5.5, 14.6	639.21	5.0, 14.6
Year-round	15712.39		15542.92 (1.08% ↓)	

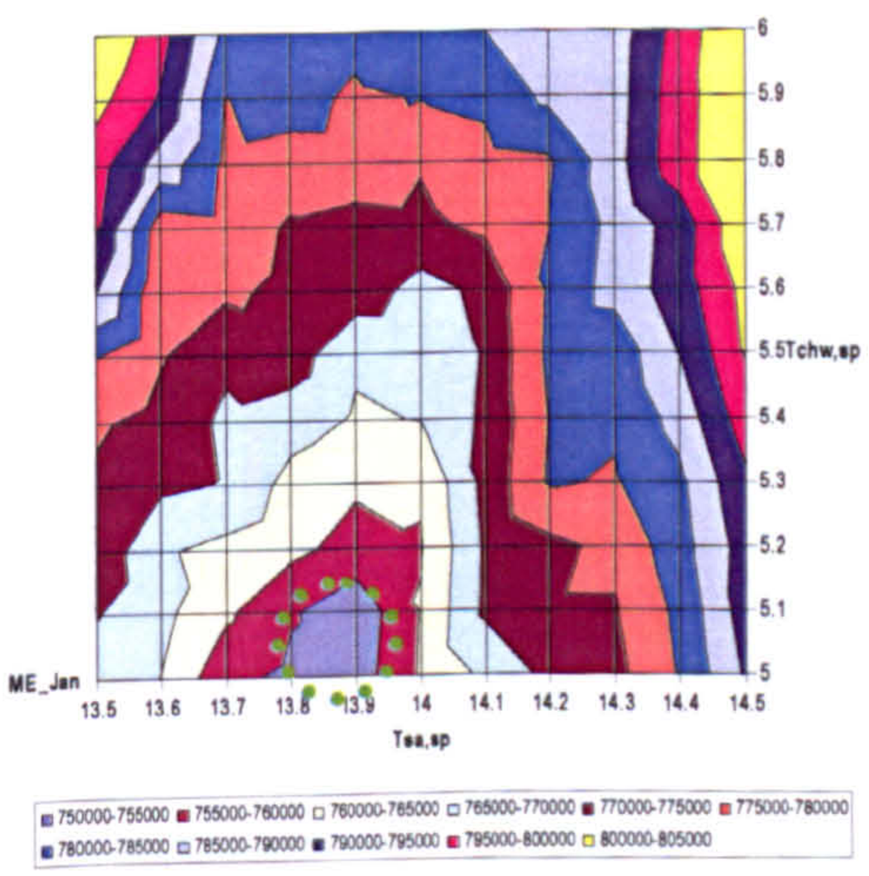


Fig. 9.13. Search contour of monthly energy consumption in January

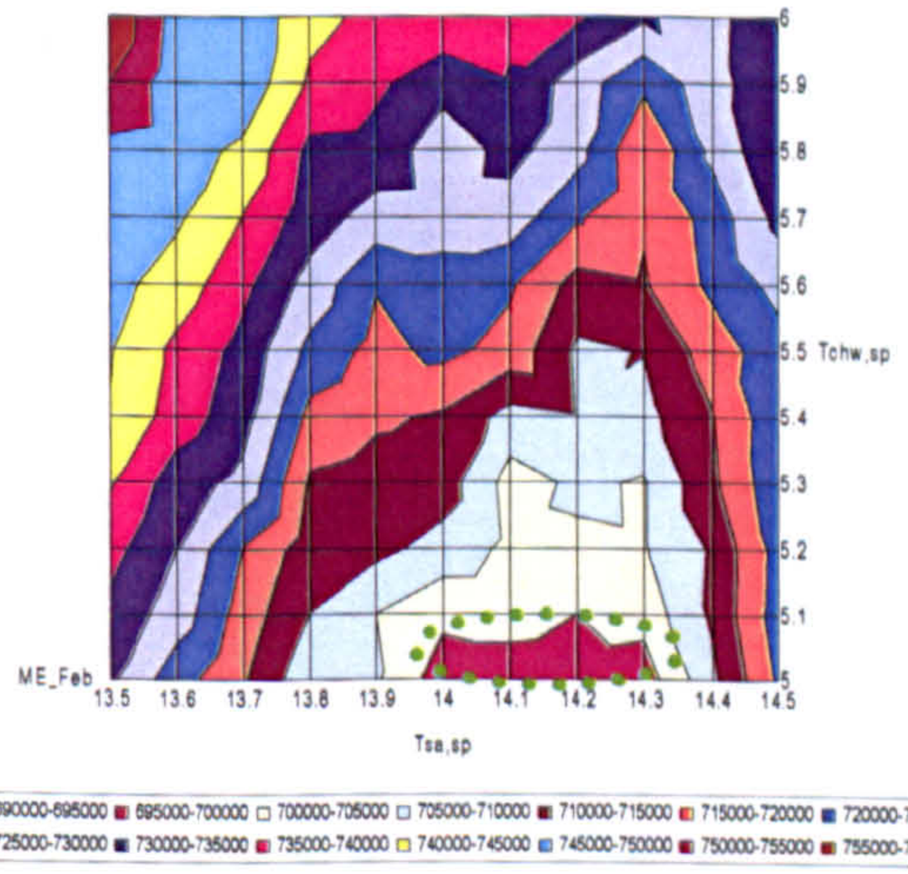


Fig. 9.14. Search contour of monthly energy consumption in February

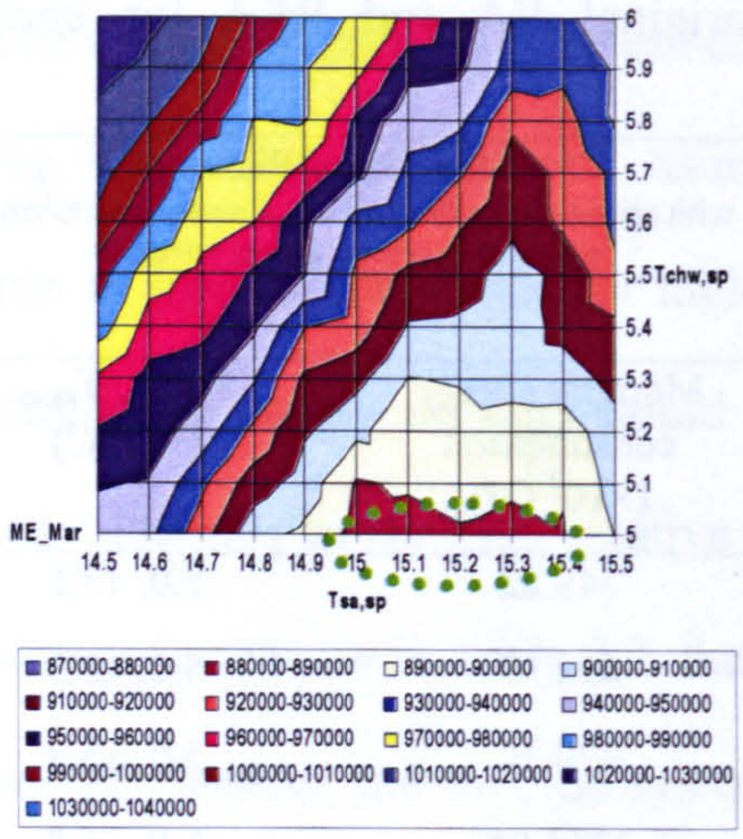


Fig. 9.15. Search contour of monthly energy consumption in March

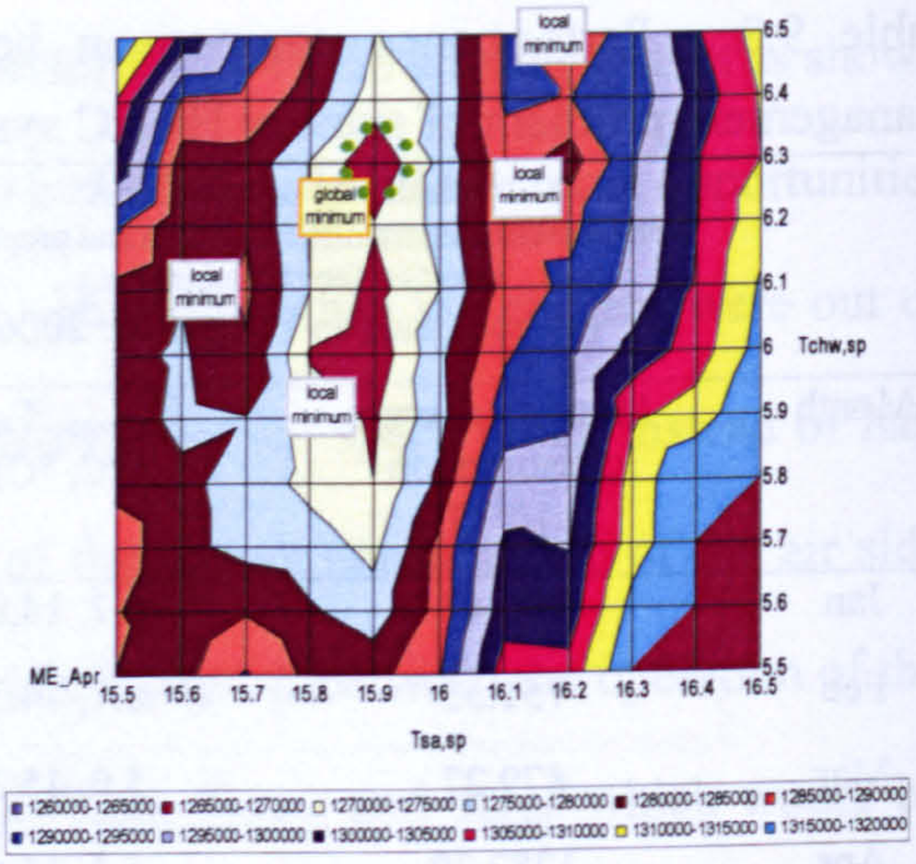


Fig. 9.16. Search contour of monthly energy consumption in April

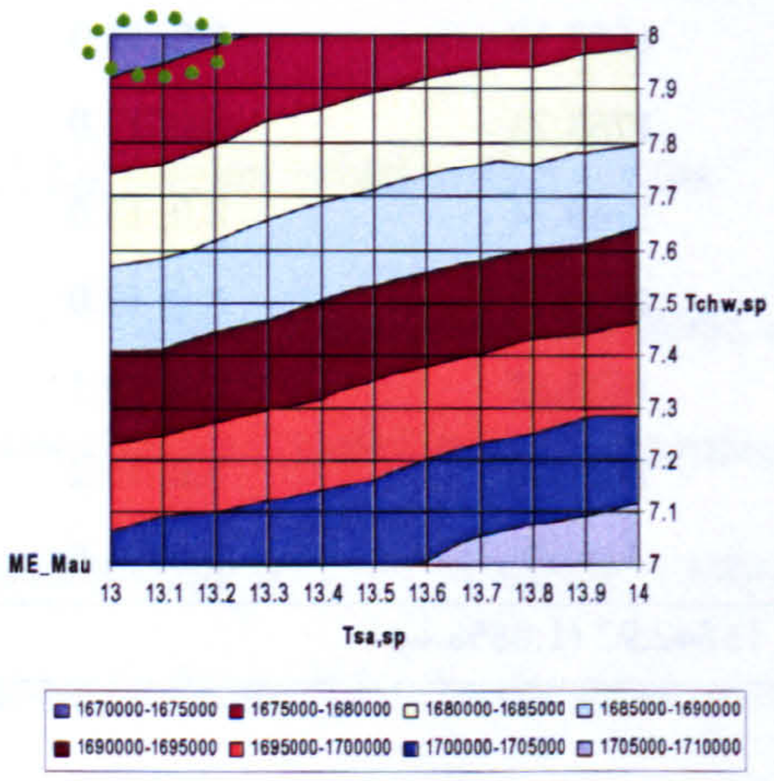


Fig. 9.17. Search contour of monthly energy consumption in May (similar to that in June, July, August and September)

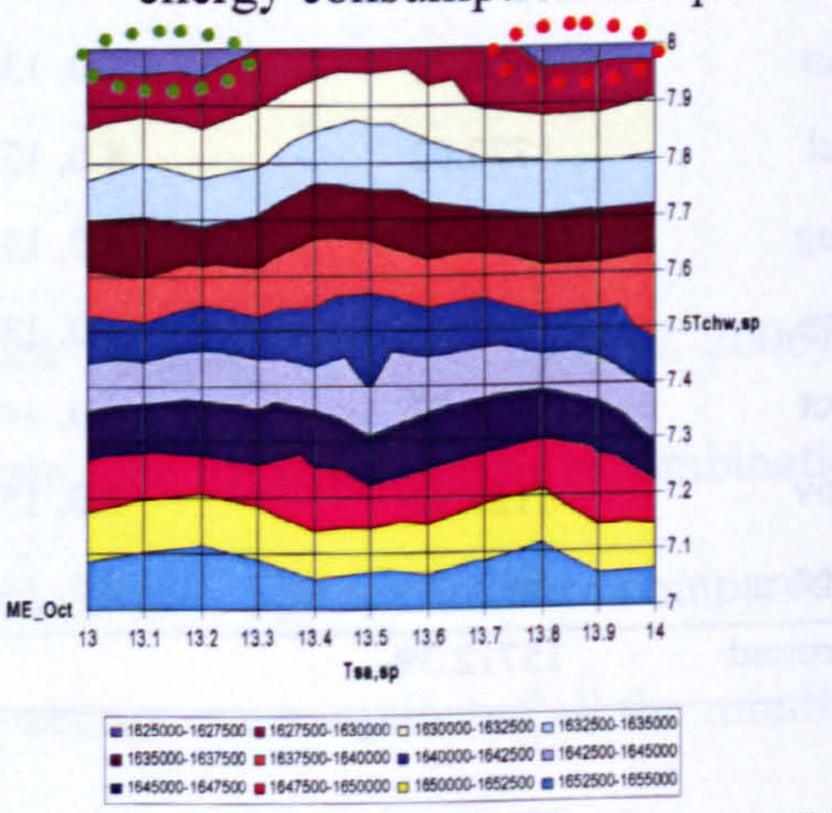


Fig. 9.18. Search contour of monthly energy consumption in October

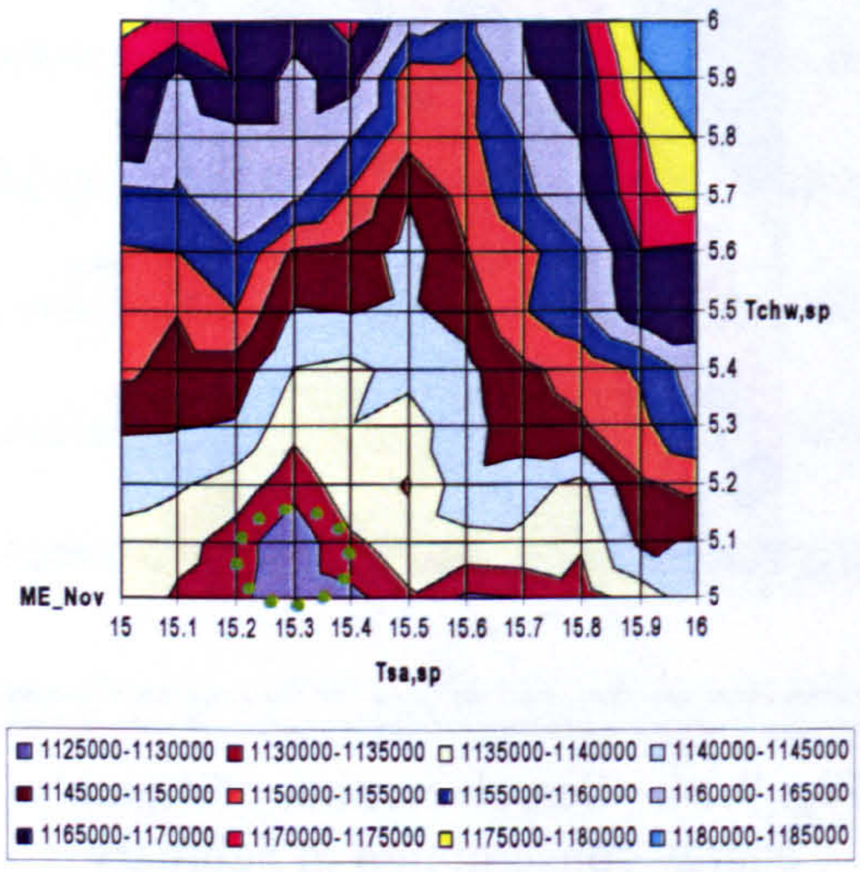


Fig. 9.19. Search contour of monthly energy consumption in November

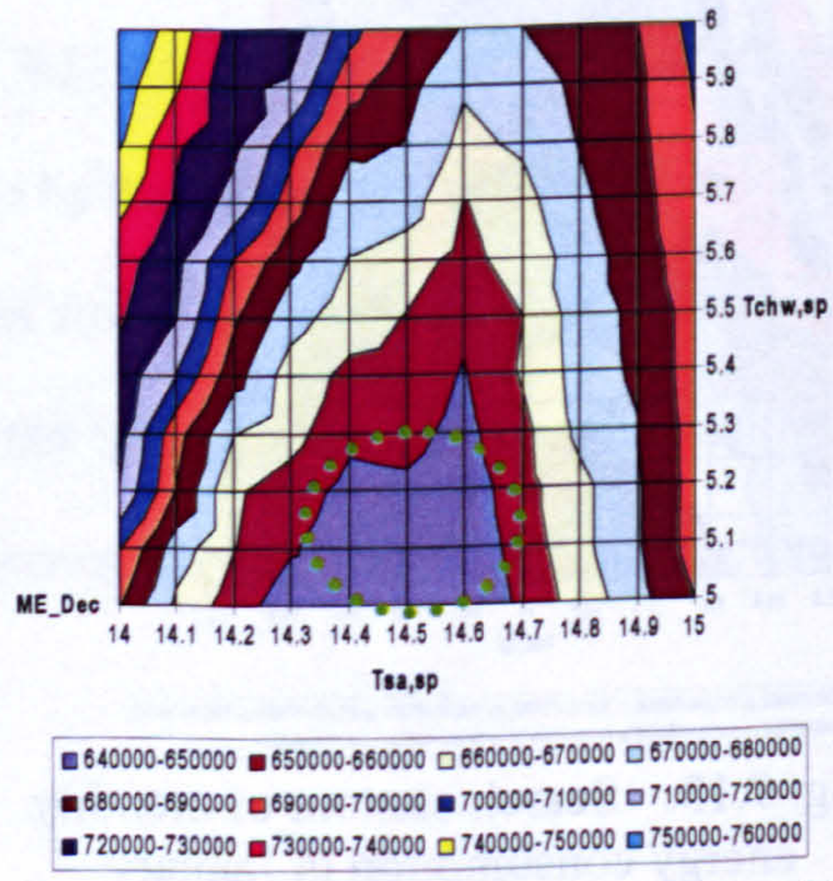


Fig. 9.20. Search contour of monthly energy consumption in December

9.2.2.5 Topography of fitness landscape

From the re-run of the unconstrained HVAC optimization problem using the REA, the results were clearly better than those previously obtained. To have a clearer explanation of this better performance, topographical analysis of the fitness landscapes was again carried out as illustrated in Figs. 9.13 to 9.20. In each figure, the possible region of the global optimum is encompassed with a green dotted line, and the local optima in red.

This problem contained a variety of topography of fitness landscapes of the monthly energy consumption throughout a year, with both unimodal and multimodal features as shown in Figs. 9.13 to 9.20. Unimodal landscapes were found in all months except April and October and the optimal search was relatively straightforward. However in April and October, multimodal landscapes were encountered, because these two months are the intermediate seasons in Hong Kong and the energy conserving control schemes for the water side and air side would be in operation occasionally as discussed in Section 9.2.2.3. Such a situation would occur more frequently in April, therefore the ruggedness of the fitness landscape was more prominent. There were four local minima around the global minimum as shown in Fig. 9.16.

9.2.3 Constrained design optimization problem – duct system

9.2.3.1 Design of duct system

Tsal *et al.* (1988) developed the T-method optimization using the paradigm of dynamic programming for a duct system design through minimization of the life-cycle cost. This method is still recommended and fully described in the ASHRAE Handbook Fundamentals 2005. There is an illustrative example of duct design, with the schematic

layout as shown in Fig. 9.21. Duct sections 1 to 6 form the return air subsystem, while duct sections 7 to 19 form the supply air subsystem. The duct lengths, fittings, accessories and other relevant design information, as well as the results of the T-method optimization, are comprehensively described in ASHRAE (2005).

Asiedu *et al.* (2000) handled this problem by using a genetic algorithm to minimize the life-cycle cost. Their study found that the GA could generate duct sizes with less life-cycle cost and better pressure balance as compared to the T-method.

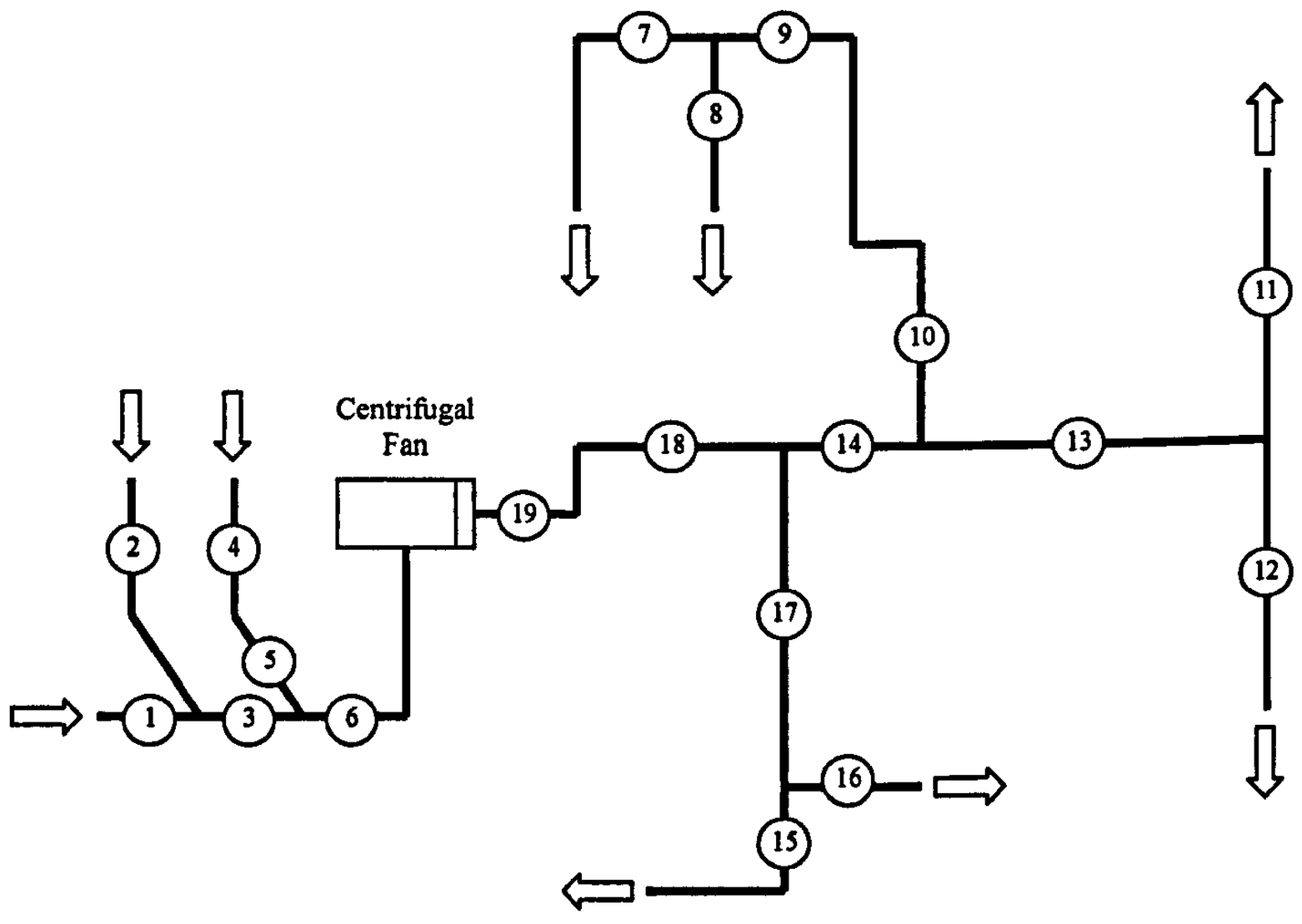


Fig. 9.21. Schematic duct layout of duct design problem in ASHRAE (2005)

9.2.3.2 Objective function

The objective function was basically the one from ASHRAE (2005), as shown in Eqs (9.7), (9.9) and (9.10). Eq (9.8) was based on that in Asiedu's paper for direct benchmarking purpose.

$$\text{minimize } F = (E_p \times PWEF) + \sum_{i=1}^{19} E_{s,i} \tag{9.7}$$

where,

F: present worth owning and operating cost ($)

$$E_p = \sum_{g=1}^{G} \frac{Q_{fan,g}(E_d + E_{c,g}\psi_g T)P_g^s}{10^3 \eta_f \eta_m}, \text{ first year energy cost ($)} \tag{9.8}$$

PWEF:present worth escalation factor

E_s $= S_d \pi L D$, initial cost for round ducts ($), or (9.9)

$= 2 S_d (H + W) L$, initial cost for rectangular ducts ($) (9.10)

D: diameter of round duct (m)

$E_{c,g}$: unit electrical energy cost in operation mode g ($/kWh)

E_d: energy demand cost ($/kWh)

G: total number of different system operation mode g

g
- = 1 for low flow and non-peak utility rate;
- = 2 for high flow and non-peak utility rate;
- = 3 for high flow and peak utility rate; or
- = 4 for low flow and peak utility rate.

H: duct height (m)

L: duct length (m)

P_g^s: maximum subsystem path pressure during operation mode g (Pa)

$Q_{fan,g}$: fan flow rate during operation mode g (m^3/s)

S_d: unit duct cost ($/m^2$)

T: operation time (hr/year)

W: duct width (m)

η_f: fan total efficiency

η_m: motor drive efficiency

ψ_g: fraction of time system operates in mode g

9.2.3.3 Constraint functions

The constraint functions were developed according to the duct design problem. There were altogether 88 constraint functions, as described below in Eqs (9.11) to (9.98).

The equality constraints C_1 and C_2 were used to limit the pressure imbalance of

the largest path, while C_3 and C_4 to limit that of all paths as follows:

$$C_1: \quad \sum_{g=1}^{G} \max_{\iota \in S_s} [\psi_g (P_g^s - P_{\iota,g})] = 0 \text{ for supply subsystem} \qquad (9.11)$$

$$C_2: \quad \sum_{g=1}^{G} \max_{\iota \in S_s} [\psi_g (P_g^s - P_{\iota,g})] = 0 \text{ for return subsystem} \qquad (9.12)$$

$$C_3: \quad \sum_{g=1}^{G} \sum_{\iota \in S_s} \psi_g (P_g^s - P_{\iota,g}) = 0 \text{ for supply subsystem} \qquad (9.13)$$

$$C_4: \quad \sum_{g=1}^{G} \sum_{\iota \in S_s} \psi_g (P_g^s - P_{\iota,g}) = 0 \text{ for return subsystem} \qquad (9.14)$$

where,

$P_{\iota,g}$: path pressure during operation mode g (Pa)

ι: duct path index

S_s: set of paths in subsystem s (i.e. supply or return subsystem)

The inequality constraints C_5 to C_{42} for air velocity limits of duct sections 1 to 19 were:

C_5 to C_{23}: $V_i - V_{mode}^{min} \geq 0$ for i = 1, 2, …, 19, mode = low or high (9.15) to (9.33)

C_{24} to C_{42}: $V_{mode}^{max} - V_i \geq 0$ for i = 1, 2, …, 19, mode = low or high (9.34) to (9.52)

where,

V: air flow velocity in duct (m/s)

The inequality constraints C_{43} to C_{87} for width control from the preceding duct section were:

C_{43}: $W_6 - W_5 \geq 0$ (9.53)

: :

C_{87}: $W_9 - W_7 \geq 0$ (9.97)

The equality constraint C_{88} for the same width of duct sections 11 and 12 (as set in the

duct design problem) were:

$$C_{88}: \qquad W_{11} - W_{12} = 0 \qquad (9.98)$$

9.2.3.4 Problem variables

The problem variables for this duct design problem included the duct sizes for the return air subsystem and those for the supply air subsystem. All these problem variables were integer I in the unit of centimetre, so every increment would be 1 cm or 10 mm. The feasible range of each problem variable was also assigned with reference to Asiedu *et al.* (2000). For the return air subsystem, there were six problem variables and their corresponding bounds were:

x_1:	diameter of duct section 1	$I \in [1, 80]$
x_2:	diameter of duct section 2	$I \in [1, 80]$
x_3:	diameter of duct section 3	$I \in [1, 80]$
x_4:	width of duct section 4 (constant, = 600 mm)	$I \in [60, 60]$
x_5:	diameter of duct section 5	$I \in [1, 80]$
x_6:	diameter of duct section 6	$I \in [1, 80]$

For the supply air subsystem, there were 13 optimization variables and their corresponding bounds were:

x_7:	width of duct section 7	$I \in [1, 80]$
x_8:	width of duct section 8	$I \in [1, 80]$
x_9:	width of duct section 9	$I \in [1, 80]$
x_{10}:	width of duct section 10	$I \in [1, 80]$
x_{11}:	width of duct section 11	$I \in [1, 80]$
x_{12}:	width of duct section 12	$I \in [1, 80]$
x_{13}:	width of duct section 13	$I \in [1, 80]$
x_{14}:	width of duct section 14	$I \in [1, 80]$
x_{15}:	width of duct section 15	$I \in [1, 80]$

x_{16}:	width of duct section 16	$I \in [1, 80]$
x_{17}:	width of duct section 17	$I \in [1, 80]$
x_{18}:	height of duct section 18	$I \in [30, 80]$
x_{19}:	width of duct section 19 (constant, = 800 mm)	$I \in [80, 80]$

9.2.3.5 Implementation and results

The results were compared between the GA of Asiedu *et al.* (2000) and the REA for the duct system design problem in Table 9.4. The optimal results were obtained from the best solution among the 10 runs reported by Asiedu *et al.* (2000). In the table, the REA can determine a lower total cost using a significant decrease of population and epoch as compared to that of the GA. Using the REA, the entire duct system with both the supply and return subsystems could be optimized together, as opposed to the GA, where the two subsystems were optimized separately. Therefore in terms of the frequency of function calls for evaluation, there was about 2600-fold reduction by using the REA.

The REA with the arithmetic recombination (XO=1) could find an optimal solution with both lower material and energy costs as compared to the GA. However, using geometrical recombination (XO=3), REA found a lower material cost but a higher energy cost, though the total cost was still lower than that from the GA. As a result, the overall performance of the REA with arithmetic recombination was better than with geometrical recombination for this problem.

Table 9.4. Comparison between GA of Asiedu *et al.* (2000) and REA for duct system design problem

	GA of Asiedu *et al.* (2000)	REA with [XO=1 + MU=2 + CH=1 + SE=2] or [XO=3 + MU=2 + CH=1 + SE=2]
Implementation		
Paradigm of evolutionary algorithm	Genetic algorithm	Evolution strategy (with $\mu=\lambda$, but using stochastic tournament selection)
Evaluation function	Eq (9.7) + penalty function as derived from the constraints of Eqs (9.11) to (9.14), incorporated with the penalty weighting parameters	Eq (9.7) only, i.e. the objective function of the problem
Constraint handling technique	Merging the penalty function with the objective function to form the fitness function, which was used to evaluate the fitness of the individuals	Handling the constraints C_1 to C_{88} with infeasibility discrimination (refer to Section 5.4.4 for detail of infeasibility discrimination)
Population size	800	10
Epoch of termination	3,359 for supply subsystem; 2,501 for return subsystem.	200
Number of function calls	4,688,000 [800 × (3359 + 2501)]	1,801 [10 + (9 × 199)]
Tournament size	5	5
Number of runs	10	10
Major Results		
Total cost ($)	11,618	11,467 [XO=1]; 11,544 [XO=3]
Material cost ($)	8,575	8,531 [XO=1]; 8,307 [XO=3]
Energy cost ($)	3,043	2,936 [XO=1]; 3,237 [XO=3]

9.2.4 Constrained energy management optimization problem – heat rejection system

9.2.4.1 Heat rejection system for centralized chiller plant

This was an optimization problem about the energy management of the heat rejection system of a centralized chiller plant as shown in Fig. 9.22.

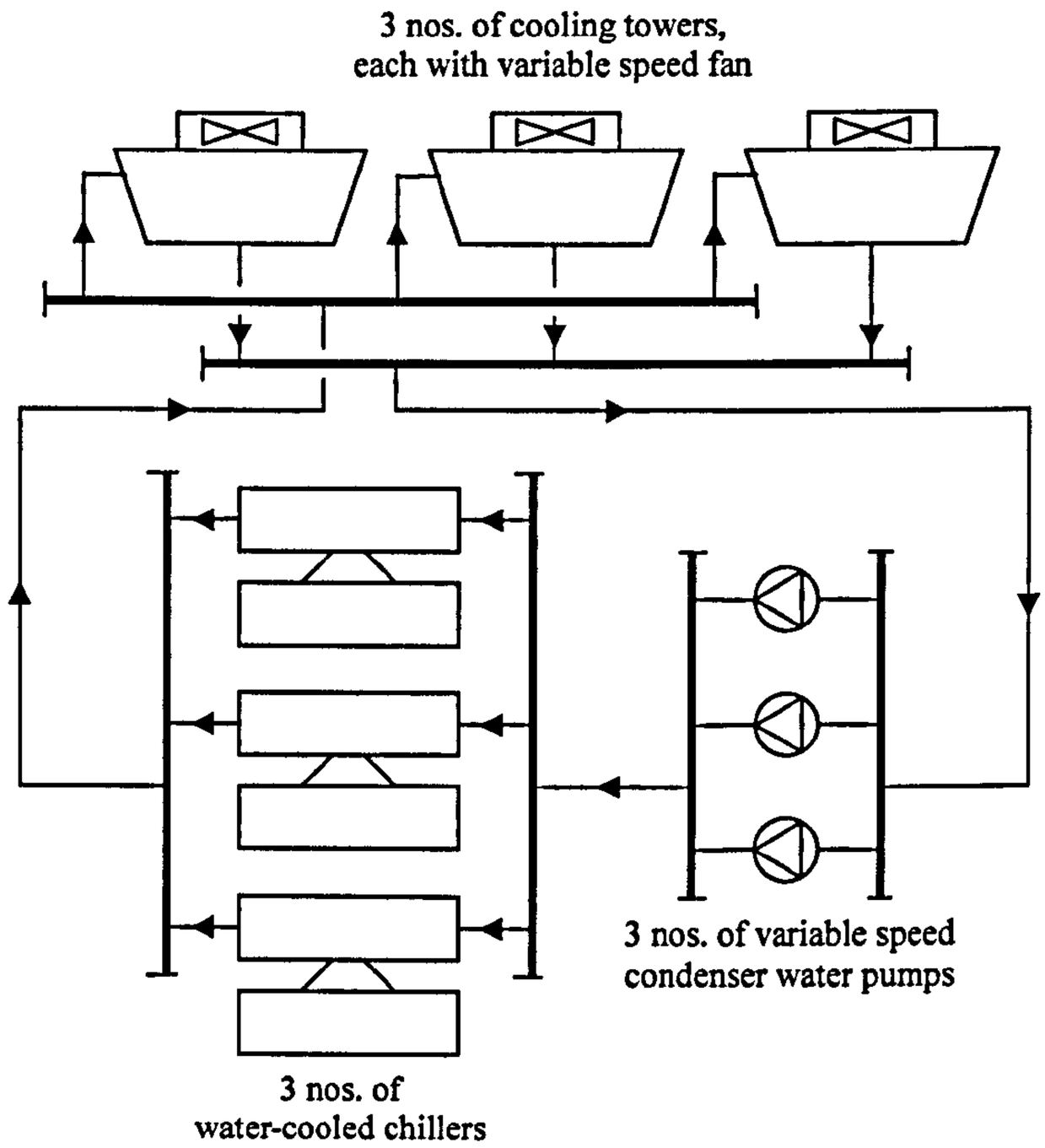


Fig. 9.22. Schematic diagram of heat rejection system of centralized chiller plant

There were three water-cooled chillers, three variable speed condenser water pumps and three cooling towers each with a variable speed fan. The goal was to minimize the daily energy consumption through optimal operation of these three components. In order to generate better schemes of energy management, the usual interlocking control between the chiller, condenser water pump and cooling tower fan was overridden, so that the deployment of these components would be more flexible. However due to practical considerations to prevent inefficient heat removal from the chiller plant, there should be constraints that the operating number of cooling tower fans would not be less than that of the condenser water pumps, and in turn the operating number of condenser water pumps not less than that of chillers.

Although the corresponding change of chilled water side and air side systems in response to the cooling load would influence the performance of the heat rejection system,

it was assumed that their operation remained intact for the elementary study of the heat rejection side. Another reason was that Lu *et al.* (2004) studied a heat rejection system using the same approach but with GA optimization, so a comparative study could be carried out with the REA directly.

9.2.4.2 Objective function

The objective of this study was to minimize the total power consumption of the entire heat rejection system in response to an hourly cooling load profile of a design day. Therefore the objective function and the related mathematical expressions were developed according to Eqs (9.99) to (9.104) below.

$$\text{minimize } F = P_{chiller} + P_{pump} + P_{fan} \tag{9.99}$$

where,

F: total power consumption of whole heat rejection loop (kW)

$$P_{chiller} = \sum_{i=1}^{n} Q_{cap,n}\, COP_{nom,n}\, (PLR_{adj,n}\, T_{adj,n}) \tag{9.100}$$

$$P_{pump} = \sum_{i=1}^{j} P_{pump,nom,j}\left[a_0 + a_1\left(\frac{m_{w,j}}{m_{w,nom,j}}\right) + a_2\left(\frac{m_{w,j}}{m_{w,nom,j}}\right)^2 + a_3\left(\frac{m_{w,j}}{m_{w,nom,j}}\right)^3\right] \tag{9.101}$$

$$P_{fan} = \sum_{i=1}^{k} P_{fan,nom,k}\left[b_0 + b_1\left(\frac{m_{a,k}}{m_{a,nom,k}}\right) + b_2\left(\frac{m_{a,k}}{m_{a,nom,k}}\right)^2 + b_3\left(\frac{m_{a,k}}{m_{a,nom,k}}\right)^3\right] \tag{9.102}$$

$$PLR_{adj,n} = c_0 + c_1\left(\frac{Q}{Q_{cap,n}}\right) + c_2\left(\frac{Q}{Q_{cap,n}}\right)^2 \tag{9.103}$$

$$T_{adj,n} = d_0 + d_1\, T_{CHWS} + d_2\, T_{CHWS}^2 + d_3\, T_{CWS} + d_4\, T_{CWS}^2 + d_5\, T_{CHWS} T_{CWS} \tag{9.104}$$

where,

$COP_{nom,n}$: nominal coefficient of performance of the n^{th} chiller

j: number of operating condenser water pump, j = 1, 2 or 3

k: number of operating cooling tower fan, k = 1, 2 or 3

$m_{a,k}$: mass flow rate of the k^{th} cooling tower fan (kg/s)

$m_{a,nom,k}$: nominal mass flow rate of the k^{th} cooling tower fan (kg/s)

$m_{w,j}$: mass flow rate of the j^{th} condenser water pump (kg/s)

$m_{w,nom,j}$: nominal mass flow rate of the j^{th} condenser water pump (kg/s)

n: number of operating chiller, n = 1, 2 or 3

$P_{fan,nom,k}$: nominal power of the k^{th} cooling tower fan (kW)

$P_{pump,nom,j}$: nominal power of the j^{th} condenser water pump (kW)

$PLR_{adj,n}$: adjustment factor for partial load ratio of the n^{th} chiller

Q: cooling load (kW)

$Q_{cap,n}$: cooling capacity of the n^{th} chiller (kW)

$T_{adj,n}$: adjustment factor for temperature of the n^{th} chiller

T_{CHWS}: chilled water supply temperature (°C)

T_{CWS}: condenser water supply temperature (°C)

The coefficients a_0 to a_3, b_0 to b_3, c_0 to c_2 and d_0 to d_5 were determined with reference to the performance diagrams of Lu *et al.* (2004), as well as the related equipment catalogues and information, so that there would be similar basis in performances for later benchmarking purpose. The values of these coefficients were determined as follows:

$a_0 = 0.0017$, $a_1 = -0.0356$, $a_2 = 0.1379$, $a_3 = 0.8656$

$b_0 = 0.0017$, $b_1 = -0.0356$, $b_2 = 0.1379$, $b_3 = 0.8656$

$c_0 = 0.0470$, $c_1 = 0.5819$, $c_2 = 0.3711$

$d_0 = -2.2097$, $d_1 = 0.0130$, $d_2 = -0.0015$, $d_3 = 0.1720$, $d_4 = -0.0018833$, $d_5 = -0.0013$

9.2.4.3 Constraint functions

There were altogether nine constraint functions in this optimization problem, as shown in Eqs (9.105) to (9.113). The first constraint was for the cooling tower with

reference to Lu *et al.* (2004), as shown in Eq (9.105). The second constraint was the interaction constraint between the heat rejection rate from the chillers and the heat removal from the condenser water, as shown in Eq (9.106). The remaining 7 constraint functions Eqs (9.107) to (9.113) were developed to formulate a practical operation scheme for the equipment for the heat rejection system.

The equality constraint C_1 for the cooling tower set was:

$$C_1: \quad Q + P_{chiller} - \sum_{i=1}^{k} \left[\frac{e_1 \, m_{a,k}^{e_3}}{1 + e_2 \left(\frac{m_{a,k}}{m_{w,k}} \right)^{e_3}} (T_{CWR} - T_{wb}) \right] = 0 \qquad (9.105)$$

where,

T_{CWR}: condenser water return temperature (°C)

T_{wb}: wet bulb temperature of cooling air (°C)

The coefficients e_1, e_2 and e_3 were determined with reference to the catalogues and related information of cooling towers, and the values were $e_1 = 5.8489$, $e_2 = 0.2107$, $e_3 = 2.1368$.

The equality constraint C_2 for heat rejection in the chiller set was:

$$C_2: \quad Q + P_{chiller} - \sum_{i=1}^{j} m_{w,j} \, C_{pw} \, (T_{CWR} - T_{CWS}) = 0 \qquad (9.106)$$

where,

C_{pw}: specific heat capacity of water (kJ/kg^1K^{-1})

The inequality constraint C_3 for positive approach of cooling tower was:

$$C_3: \quad T_{CWS} - T_{wb} \geq 0 \qquad (9.107)$$

The inequality constraint C_4 for positive temperature difference of condenser water was:

$$C_4: \quad T_{CWR} - T_{CWS} \geq 0 \tag{9.108}$$

The inequality constraint C_5 for the number of operating cooling tower fans k not less than that of operating condenser water pumps j, was:

$$C_5: \quad k - j \geq 0 \tag{9.109}$$

The inequality constraint C_6 for the number of operating cooling tower fans k not less than that of operating chillers n, was:

$$C_6: \quad k - n \geq 0 \tag{9.110}$$

The inequality constraint C_7 for the number of operating condenser water pumps j not less than that of operating chillers n, was:

$$C_7: \quad j - n \geq 0 \tag{9.111}$$

The inequality constraint C_8 for preventing total capacity of operating chillers $Q_{cap,sum}$ from not satisfying hourly cooling load Q, where $Q_{ratio} = Q / Q_{cap,sum}$, in case that two chillers should be operated instead of only one, was:

$$C_8: \quad 1 - Q_{ratio} \geq 0 \quad \text{if } n = 2 \tag{9.112}$$

The inequality constraint C_9 for preventing total capacity of operating chillers $Q_{cap,sum}$ from not satisfying hourly cooling load Q, in case that three chillers should be operated instead of only one, was:

$$C_9: \quad 1 - Q_{ratio} \geq 0 \quad \text{if } n = 3 \tag{9.113}$$

9.2.4.4 Problem variables

There were altogether seven problem variables, the details and their corresponding bounds are listed below. One of the features of this optimization problem was to handle the problem variables in both real number R (x_1 to x_4) and integer I (x_5 to x_7) forms.

x_1:	$m_{w,j}$, flow rate of operating condenser water pump	$R \in [0.05, 0.5]$
x_2:	$m_{a,k}$, air flow rate of operating cooling tower fan	$R \in [0.02, 0.2]$
x_3:	T_{CWS}, condenser water supply temperature	$R \in [25, 45]$
x_4:	T_{CWR}, condenser water return temperature	$R \in [25, 45]$
x_5:	n, number of chillers in operation	$I \in [1, 3]$
x_6:	j, number of condenser water pumps in operation	$I \in [1, 3]$
x_7:	k, number of cooling tower fans in operation	$I \in [1, 3]$

9.2.4.5 Implementation and results

Lu *et al.* (2004) also used Eqs (9.100) to (9.104) to develop the mathematical expressions of the equipment of heat rejection system in their study, hence the same objective function in Eq (9.99) was used. In Lu's work, Eqs (9.105) and (9.106) were applied as constraint functions, but handled by a static penalty. In order to make a comparison between the results of the GA in Lu's work and those of the REA, the numerical details in Lu's paper were used. These included the capacity and features of the equipment as well as the daily cooling load profile. Lu *et al.* (2004) devised an optimal operation strategy for this heat rejection system through minimizing the total power consumption by GA.

The results from the REA were chosen from the best feasible individual within the 10 runs. The comparison between GA of Lu *et al.* (2004) and the REA for this problem are summarized in Table 9.5. Although the results from the REA were better than those

of GA in Lu *et al.* (2004), these results could not be compared directly, since the coefficients and some input information were not clearly defined in Lu's paper. The results of the hourly total power profile and the hourly operation of different components of the heat rejection system are shown in Fig. 9.23 and Table 9.6 respectively.

Table 9.5. Comparison between GA of Lu *et al.* (2004) and REA for energy management problem of HVAC heat rejection system

	GA of Lu *et al.* (2004)	REA with [XO=1 + MU=2 + CH=1 + SE=2] or [XO=3 + MU=2 + CH=1 + SE=2]
Implementation		
Paradigm of evolutionary algorithm	Genetic algorithm	Evolution strategy (with $\mu=\lambda$, but using stochastic tournament selection)
Evaluation function	Eq (9.99) + penalty function as derived from the constraints of Eqs (9.105) and (9.106), incorporated with the penalty weighting parameters	Eq (9.99) only, i.e. the objective function of the problem
Constraint handling technique	Merging the penalty function with the objective function to form the fitness function, which was used to evaluate the fitness of the individuals	Handling the constraints C_1 to C_9 with infeasibility discrimination (refer to Section 5.4.4 for details of infeasibility discrimination)
Population size	100	10
Epoch of termination	500	400
Number of function calls	50,000 (100 × 500)	3,601 (10 + 9 × 399)
Number of runs	Not mentioned	10
Major Results		
Daily total power (kW)	48.42	45.45 [XO=1]; 48.35 [XO=3]

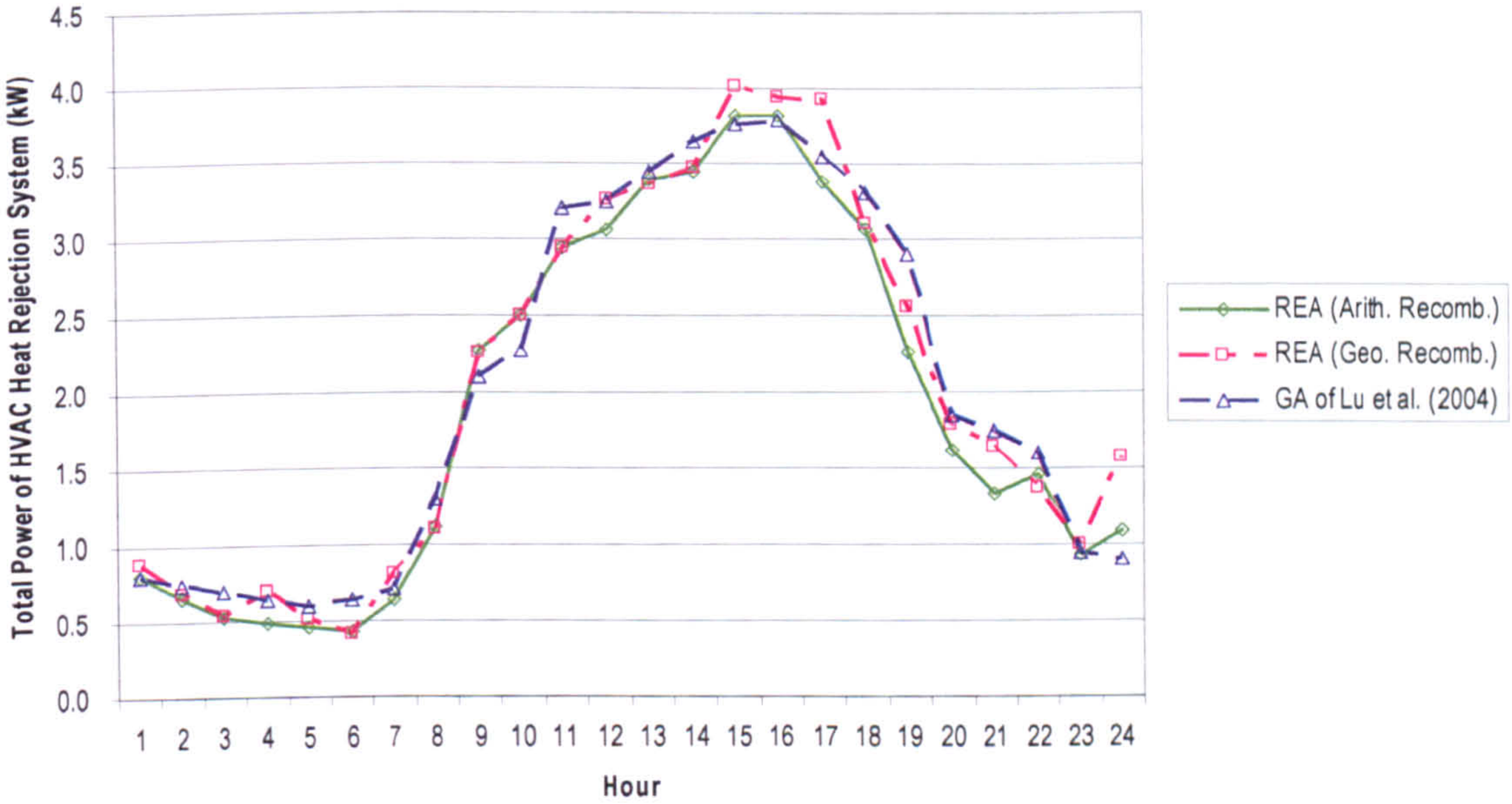


Fig. 9.23. Hourly total power profile of HVAC heat rejection system

Table 9.6. Quantity of operating equipment and hourly total power of HVAC heat rejection system

	Published Results in Lu *et al.* (2004) by Using GA				Optimization Results by REA with (Arithmetic Recombination)				Optimization Results by REA with (Geometrical Recombination)			
Hour	n	j	k	P_{total} (kW)	n	j	k	P_{total} (kW)	n	j	k	P_{total} (kW)
1	1	1	3	0.80	1	1	3	0.800	1	1	3	0.882
2	1	1	3	0.75	1	1	3	0.664	1	1	3	0.684
3	1	1	3	0.70	1	1	3	0.530	1	1	3	0.550
4	1	1	3	0.65	1	1	3	0.485	1	1	3	0.702
5	1	1	3	0.60	1	1	3	0.454	1	1	3	0.511
6	1	1	3	0.65	1	1	3	0.431	1	1	3	0.412
7	1	1	3	0.72	1	1	3	0.643	1	1	3	0.811
8	2	3	3	1.30	2	3	3	1.116	2	2	3	1.102
9	2	3	3	2.10	2	3	3	2.266	2	3	3	2.254
10	2	3	3	2.27	2	3	3	2.503	2	3	3	2.502
11	3	3	3	3.20	3	3	3	2.950	3	3	3	2.939
12	3	3	3	3.25	3	3	3	3.057	3	3	3	3.259
13	3	3	3	3.45	3	3	3	3.380	3	3	3	3.358
14	3	3	3	3.65	3	3	3	3.437	3	3	3	3.468
15	3	3	3	3.76	3	3	3	3.819	3	3	3	4.008
16	3	3	3	3.78	3	3	3	3.820	3	3	3	3.937
17	3	3	3	3.55	3	3	3	3.375	3	3	3	3.933
18	3	3	3	3.30	3	3	3	3.057	3	3	3	3.106
19	3	3	3	2.90	3	3	3	2.251	3	3	3	2.563
20	2	3	3	1.85	2	3	3	1.610	2	3	3	1.793
21	2	3	3	1.74	2	3	3	1.329	2	3	3	1.644
22	2	3	3	1.60	2	2	3	1.452	2	2	3	1.371
23	1	2	3	0.95	1	2	3	0.935	1	2	3	0.998
24	1	2	3	0.90	1	1	3	1.088	1	2	3	1.566
			Total	48.42			Total	45.45			Total	48.35

From Tables 9.5 and 9.6, the daily total power of the REA with geometrical recombination (XO=3) was very close to that of the GA. However from Fig. 9.23, the profile of the former around the peak was slightly higher than that of the latter. On the other hand, the daily power of the REA with arithmetic recombination (XO=1) was slightly less than that of the GA, but its profile was close to that of the GA. As a whole for the REA, whichever recombination was used, the population and epoch were just 10 and 400 respectively, but those of the GA were higher at 100 and 500 respectively. Therefore the REA had about a 14-fold reduction of the evaluation function calls as compared to the GA, hence the execution time was significantly decreased. As a result, the REA showed much better efficiency than the GA of Lu's work.

By comparing the daily total power generated from the REA with arithmetic recombination (45.45 kW) to that with geometrical recombination (48.35 kW), it was found that the former gave a 6% reduction against the latter. This showed again that the arithmetic recombination had a better performance and higher potential to search for optimal results, as also found in the constrained problem of duct system design in Section 9.2.3.5.

9.3 Summary

Through the application of the REA in the HVAC optimization problems of system design and energy management, better results were found, both in terms of effectiveness and efficiency. In the unconstrained problems of solar water heating and subway HVAC system, better performance of the REA was found as compared to the original EA (Gaussian deterministic mutation + proportional selection). From the topographical study, the multimodal landscape around the global optimum could be more

effectively tackled by the REA. For the constrained problems of duct design and heat rejection system, both effectiveness and efficiency of the REA could be enhanced as benchmarked with the published results using GA (Asiedu 2000 and Lu *et al.* 2004 respectively) to handle the same HVAC problems. Particularly the significant reduction of evaluation function calls showed the outstanding performance of the REA in optimization efficiency.

For the four HVAC optimization problems under study, the computational cost of evaluation function calls in respect of the overall running time are summarized in Table 9.7 below.

Table 9.7. Computational cost of evaluation function calls of the four HVAC optimization problems under study

HVAC Optimization Problem	Problem Nature	Number of Population n_{pop}	Epoch of Termination $epoch_{max}$	Number of Evaluation Function Calls n_{eval}	Running Time of Each Evaluation Call (s)	Percentage of Running Time of Evaluation
System design of solar water heating	Unconstrained design	10	50	451	218.68	99.98%
Energy management of subway station	Unconstrained energy management	10	50	451	2.09	99.70% (average of Jan to Dec)
Design of duct system	Constrained design	10	200	1801	0.49	95.84%
Energy management of heat rejection system	Constrained energy management	10	400	3601	1.11×10^{-3}	16.32%

Remark: The number of evaluation function calls was determined from Eq (5.32), and repeated as follows:

$$n_{eval} = epoch_{max} (n_{pop} - 1) + 1$$

From Table 9.7, in view of the overall computational time of optimization-simulation run,

the percentage was generally greater than 95% (except the fourth problem). This implied that the running time of evaluation function calls was nearly same as the computational period for the entire optimization process. The fourth HVAC optimization problem was an exception, since the plant models of power consumption were simplified for real time application (Lu *et al.* 2004), and represented by the objective and constraint functions that could be directly related to the problem variables (Eqs (9.100) to (9.105)), hence no iteration nor manipulation of complex system of equations was involved. Therefore the evaluation process of the fourth problem was very straightforward, the required computational time would be normal and not dominated by the evaluation process.

In handling the two constrained HVAC optimization problems, arithmetic recombination had better performance than geometrical recombination. This showed that the REA with arithmetic recombination could provide good optimal solutions in general, although geometrical recombination could give superior results for certain specific forms of function. Therefore, the synergetic combination of the REA with Cauchy deterministic mutation, tournament selection and arithmetic recombination provides promising results for the HVAC optimization problems.

CHAPTER 10 CONCLUSION AND FUTURE WORK

In the optimized design and energy management of heating, ventilating and air conditioning (HVAC) systems, solving a large set of linear and nonlinear differential-algebraic equations would commonly be involved in the detailed simulation models. It is crucial to use an effective optimization method that the required number of simulation runs or evaluation function calls can be minimized, and the optimal solution can still be satisfactorily determined. Owing to the nonlinear and mixed real-discrete nature of HVAC optimization problems, traditional methods become inappropriate, and suitable numerical or heuristic approach should be adopted. After careful study, evolutionary algorithm (EA) is found to be effective in handling HVAC optimization problems. Through a series of systematic analyses and benchmarking exercises in this research, the EA Suite optimization platform has been established upon the paradigms of evolution strategy and evolutionary programming, instead of genetic algorithm (GA). In addition, the robust EA (REA) has been formulated, with outstanding effectiveness and efficiency in global search at minimum evaluation function calls. A robust coupling linkage between the EA Suite and the plant simulation program TRNSYS has also been devised. All these outcomes are in line with the three research objectives stated in Section 1.3. The entire research works are summarized below, and the major contributions of this research are highlighted. The direction of future work is also explored.

10.1 Summary of Research Works

Simulation model for complete HVAC optimization problem. A plant simulation model for typical centralized HVAC optimization problems has been developed. It is

component-based, with a dynamic operation algorithm to call in the required equipment in response to changing hourly cooling loads and outdoor conditions throughout a year.

EA optimization platform. A modular EA optimization platform, EA Suite, has been established based on the paradigms of evolution strategy and evolutionary programming. The core EA operators include mutation, selection, recombination and constraint handling. A number of options for each EA operator are available, so that the required combination of EA operators can be set. On this modular platform, an alternative evolutionary optimization method, micro-genetic algorithm (MGA), has also been developed.

EA Suite with coupling provision. The EA Suite includes a coupling linkage to the TRNSYS plant simulation program. The EA evaluation process for any new set of problem variables can be run via the plant simulation model through the TRNSYS console engine version 15.3. The output from the simulation model gives the corresponding fitness values to be passed back to the EA platform. This fitness information is then used in the subsequent stages of EA optimization.

Development of new EA operators. Three new EA operators have been devised in this research, they are Gaussian deterministic mutation, Cauchy deterministic mutation, and infeasibility discrimination constraint handling operators. The Cauchy deterministic mutation is the core operator of the REA, it can effectively provide the explorative direction and step length through the joint effect of Cauchy realization and deterministic strategy parameter.

Major results from comparative studies. In this research work, comparative study was the major approach used to analyze the performance of different operators and combinations in the EA Suite. Comparative studies were applied in the following areas

and the corresponding research outcomes are summarized as follows:

- *Formulation of REA.* A wide range of experiments was conducted to study the contributions of different combinations of EA operators through a series of test functions. It was found that the best core EA operators were Cauchy deterministic mutation and tournament selection, with enhancement by arithmetic recombination. These three EA operators formed the basis of the proposed REA in this thesis. For constrained problems, infeasibility discrimination was recommended as the constraint handling operator due to its generally stable performance.
- *Benchmarking with MGA.* Benchmarking was carried out by comparing the REA with the well-documented optimization method MGA. It was found that the overall performance of the REA was better than that of MGA for a variety of constrained and unconstrained test problems on the same basis of a limited population of five.
- *Contribution of constraint handling techniques.* Another series of comparative studies were conducted to study the effect of implementing different constraint handling techniques in the EA Suite. Comparisons were made against results from the TS-R method (Deb 2000) and the non-dominance method (Coello 2002). It was found that under the rank-based constraint handling approach developed in the EA Suite, the contribution of different constraint handling operators in the optimization process was not as important as what has been reported in the case of the GA. In fact, the major contribution to satisfactory performance comes from the core operators of mutation, selection and recombination of REA, the constraint handling operator is rather a background operator once a feasible elite individual is identified.

Applications in HVAC optimization problems. The developed REA was applied to both unconstrained and constrained HVAC design and energy management

optimization problems. All the results demonstrated that the REA could give effective and promising performance. In the cases of duct design and heat rejection system, the performances of the REA were also benchmarked with those of Asiedu *et al.* (2000) and Lu *et al.* (2004) respectively. It was found that the REA was able to generate better optimal results, and achieve a significant reduction in evaluation function calls.

10.2 Contributions of the Research

The major contributions of this research cover the following areas:

a. *Robust optimization method.* The REA has been found to be an effective and efficient optimization method to solve HVAC optimization problems with a multimodal, multidimensional, nonlinear, mixed discrete-continuous, and highly constrained nature. Such complex search landscapes can contain many local optima which would readily trap some optimization methods. However the REA has been shown to be capable of overcoming this problem and can determine the global or near optimum.

b. *Economy of evaluation function calls.* The default population of the REA is only ten, which is far less than that commonly used in the paradigm of GA. Usually the population of a GA would be in tens or hundreds in order to maintain the diversity of search and the contribution of elite individuals. The reason that REA can provide good performance with small population is due to the performance of the core EA operators involved. Cauchy deterministic mutation, tournament selection and arithmetic recombination can have a synergetic effect in both exploration and exploitation. In the REA, the inclusion of elitism further reduces the number of evaluation function calls to nine from the second epoch onwards, since the evaluated

fitness of the elite individual is already in memory.

c. *Better optimization performance from paradigm of evolution strategy.* The structure of the REA is related to the paradigm of evolution strategy. Since the contribution of a recombination has been shown to be beneficial, this algorithm has a better performance than evolutionary programming. The reliability of determining the global or near optimum is higher in the case of the REA, in particular with regard to the number of evaluation function calls.

d. *Simple problem-independent implementation.* No problem-specific parameters are needed in the REA. Domain-specific knowledge is generally not required (except when using geometrical recombination). The configuration is simple and straightforward, but its performance is better or comparable to other EAs with specially designed operators (e.g. MGA, GA with the TS-R or non-dominance constraint handling method).

e. *Effective with both constrained and unconstrained problems.* The REA, with the core operators of Cauchy deterministic mutation, tournament selection and arithmetic recombination, can handle both the constrained and unconstrained optimization problems effectively. The constraint handling technique of infeasibility discrimination is used to tackle constrained problems.

f. *Independent to choice of constraint handling operators.* The choice of constraint handling techniques is not critical in the EA Suite. Generally the search by REA can identify a feasible elite individual within the initial 5% of epoch of termination, no matter which kind of constraint handling operators are used. The constraint handling technique thus plays its role only at the initial stage of the optimal search in REA.

g. *Possible applications outside the HVAC field.* It would be possible to extend the applications of this REA to other optimization problems with similar nature of HVAC problems. Therefore there is potential of the REA to be applied in other engineering and mechanical optimization problems.

10.3 Future Work

Further development of the EA Suite. The developed EA Suite is a prototype, and it is a good foundation for continual improvement and advancement. The future development could include the following aspects:

- Apart from the current plant simulation program, the external coupling link may be extended to building energy simulation programs such as EnergyPlus or DOE-2, so that the EA Suite can also be applied to optimization models related to whole building energy simulation.
- Newly emerging EA operators of mutation, selection and recombination can be added to the platform of the EA Suite and their performances evaluated. Therefore more options of EA operators could be available in the future.

Analytical study of REA. The analytical study of the real-valued EA is a challenge to researchers in the field of theoretical computer science. He and Yao (2003) have established a unified framework for the analytical study of the computational time (first hitting time) of (1+1) and population-based EA, but the framework was still based on the binary representation. Bienvenüe and François (2003) have developed a more rigorous theoretical framework to prove the global convergence of evolution strategy. A simplified induced Markov chain was applied in the mathematical proof. However the

study was limited to the (1, λ) evolution strategy, and the proof was applied to the Sphere Model only, hence is not directly transferable to the other problems. Also by using Markov chain, Yuen and Cheung (2006) have developed an approximate aggregated finite population model that can be used to derive the lower and upper bounds for the expected probability of the global optimal solution for the generalized unitation functions, and only limited to the binary coded GA using proportional selection and uniform crossover, uniform mutation without elitism.

The determination of the required transition probabilities of the related Markov chain is not straightforward. Conceptually, this transition probability is defined as the probability of becoming a state with a better individual in the current epoch given a state of individual in the preceding epoch. It is possible to determine the transition probability if (1, λ)-EA based on deterministic selection. If the scenario becomes population-based with stochastic selection, like the current REA in this thesis, the situation would become very complex. As a result, to carry out an analytical study of the REA for HVAC optimization problems would not be presently feasible. Therefore, continual research work on optimized design and energy management of HVAC systems will be based on the methodologies of empirical studies and qualitative analysis. However the *analytical study* of EA is a worthwhile direction and this will lead to more theoretical knowledge to be built up.

References

Angelov, P.P., Zhang, Y., Wright, J.A., Hanby, V.I., Buswell, R.A., 2003. Automatic design synthesis and optimization of component-based systems by evolutionary algorithms. GECCO 2003, LNCS 2724, pp.1938-1950.

ASHRAE 2005. ASHRAE Handbook Fundamentals 2005, Ch. 32, Example 6. American Society of Heating, Refrigerating and Air-conditioning Engineers, Inc.

Asiedu, Y., Besant, R.W. and Gu, P., 2000. HVAC Duct System Design Using Genetic Algorithms. HVAC&R Research, 6(2), April 2000.

Bäck, T. and Schwefel, 1993. An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation, 1(1) pp.1-23.

Bazaraa, M.S., Sherali, H.D., Shetty, C.M., 1993. Nonlinear Programming: Theory and Algorithms, second ed. Wiley, New York.

Bienvenüe, A. and François, O., 2003. Global convergence for evolution strategies in spherical problems: some simple proofs and difficulties. Theoretical Computer Science 306 pp.269-289.

BLAST 1993. BLAST Technical Reference, BLAST Support Office, University of Illinois at Urbana-Champaign.

Box, M.J., 1965. A new method of constrained optimization and a comparison of other methods. The Computer Journal 8 pp.42-52.

Braun, J.E., 1988. Methodologies for the design and control of chilled water systems, Ph.D. Thesis, University of Wisconsin - Madison.

Bremermann, H.J., 1962. Optimization through evolution and recombination. Self-Organizing Systems, ed. Yovits M.C. *et al.* (Washington, DC: Spartan)

Bremermann, H.J., Rogson, M. and Salaff, S., 1965. Search by evolution. Biophysics and Cybernetic Systems – Proceedings of 2nd Cybernetic Sciences Symposium, ed. Maxfield, M., Callahan, A. and Fogel, L.J. (Washington, DC: Spartan) pp.157-167.

Bris Data AB, 1999. IDA Simulation Environment – User's Manual. Bris Data AB, Västerlånggatan 27, Stockholm, Sweden.

Broyden, G.C., 1965. A class of methods for solving nonlinear simultaneous equations. Mathematics of Computation. 19 pp.577-593.

Chan, A.L.S., Chow, T.T., Fong S.K.F., Lin J.Z., 2006. Generation of a typical meteorological year for Hong Kong. Energy Conversion and Management 47 pp.87-96.

Chen, T.Y., 2001. Real-time predictive supervisory operation of building thermal systems with thermal mass. Energy and Buildings 33 pp.141-150.

Chow, T.T., Fong, K.F., Chan, A.L.S., Lin, Z., 2006. Potential application of a centralized solar water-heating for a high-rise residential building in Hong Kong. Applied Energy 83 pp.42-54.

Chow, T.T., Zhang, G.Q., Lin, Z., Song, C.L., 2002. Global optimization of absorption chiller system by genetic algorithm and neural network. Energy and Building 34 pp.103-109.

Clarke, J.A., 1985. HVACSIM+ Building Systems and Equipment Simulation Program – Reference Manual, National Institute of Standards and Technology, Washington D.C.

Coello, C.A.C., 2000a. Constraint-handling using an evolutionary multiobjective optimization technique. Civil Engineering and Environmental Systems 17 pp.319-346.

Coello, C.A.C., 2000b. Use of a self-adaptive penalty approach for engineering optimization problems. Computers in Industry 41(2) pp.113-127.

Coello, C.A.C., 2002. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer methods in applied mechanics and engineering 191 pp.1245-1287.

CPPLD 2000. Code of Practice for Prevention of Legionnaires' Disease, Prevention of Legionnaires' Disease Committee, Government of the Hong Kong Special Administrative Region, 2000.

Dasgupta, D., 1997. Optimal scheduling of thermal power generation using evolutionary algorithms. Evolutionary Algorithms in Engineering Applications ed. D. Dasgupta and Z. Michalewicz (Berlin: New York: Springer) pp.317-328.

De Jong, K.A., 1975. An analysis of the behaviour of genetic adaptive systems. Dissertation Abstracts International 41(9) 3505B.

Deb, K., 1991. Optimal design of a welded beam via genetic algorithms, AIAA J. 29(11) pp.2013-2015.

Deb, K., 1997. GeneAS: A robust optimal design technique for mechanical component design, in: D. Dasgupta, Z. Michalewicz (Eds.), Evolutionary Algorithms in Engineering Applications, Springer, Berlin, pp.497-514.

Deb, K., 2000. An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering 186 pp.311-338.

Dong, J., Shi, J., Wang, S., Xue, y., Liu, S., 2003. A trust-region algorithm for equality-constrained optimization via a reduced dimension approach. Journal of Computational and Applied Mathematics 152 pp.99-118.

Dorigo, M. and Maria, G., 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transaction Evolutionary Computation 1 pp.53-66.

EnergyPlus 2006. EnergyPlus Engineering Reference, The Reference to EnergyPlus Calculations, September 22, 2006.

Fletcher, R., 1963. Generalized inverses for nonlinear equations and optimization. In Rabinowitz, R. (ed.), Numerical Methods for Non-linear Algebraic Equations. London: Gordon and Breach.

Fletcher, R., 1987. Practical Methods of Optimization, second ed. Wiley, Chichester.

Floundas, C. and Pardalos, P., 1987. A Collection of Test Problems for Constrained Global Optimization, volume 455 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany.

Fogel, D.B., 1988. An evolutionary approach to the traveling salesman problem. Biological Cybernet 60 pp.139-44.

Fogel, D.B., 1992. Evolving Artificial Intelligence, PhD Thesis, University of California, San Diego.

Fogel, D.B. and Fogel, L.J., 1988. Route optimization through evolutionary programming, Proceedings on 22nd Asilomar Conference on Signals, Systems and Computers (Pacific Grove, CA) pp.679-680.

Fogel, L.J., 1962. Autonomous automata. Industrial Research 4 pp.14-19.

Fogel, L.J., 1964. On the Organization of Intellect, PhD Thesis, University of California at Los Angeles.

Fong, K.F., Chow, T.T., Chan, L.S., Ma, W.L., Fong, C.K., 2001. A preliminary study of optimization of pipe route design of district cooling system. Proceedings of the 4th International Conference on Indoor Air Quality, Ventilation and Energy Conservation in Buildings, Vol. II, pp. 1031-1038, China.

Fong, K.F., Chow, T.T., Hanby, V.I., 2005a. Development of optimal design of solar water heating system by using evolutionary algorithm. Journal of Solar Energy Engineering, submitted for review.

Fong, K.F., Hanby, V.I. and Chow, T.T., 2003. Optimization of MVAC systems for energy management by evolutionary algorithm. Facilities, 21(10) pp.223-232.

Fong, K.F., Hanby, V.I. and Chow, T.T., 2004. Optimal energy management of HVAC systems by using evolutionary algorithm. Proceedings of the CIB World Building Congress 2004 Building for the Future, Toronto, Canada, May 2004, paper 737 (CD-ROM)

Fong, K.F., Hanby, V.I. and Chow, T.T., 2005b. Application of evolution strategies for HVAC optimization problem. Proceedings of the 2005 World Sustainable Building Conference, Tokyo, Japan, September 2005.

Fong, K.F., Hanby, V.I., Chow, T.T., 2006. HVAC system optimization for energy management by evolutionary programming. Energy and Buildings 38 pp.220-231.

Fraser, A.S., 1957. Simulation of genetic systems by automatic digital computers. Aust Journal of Biological Science 10 pp.484-499

Furey, B.P., 1993. A sequential quadratic programming-based algorithm for optimization of gas networks. Automatica 29(6) pp.1439-1450.

GAT 2006. Genetic Algorithm Toolbox, Evolutionary Computation Research Group, Department of Automatic Control and Systems Engineering, University of Sheffield 2005. Internet web site: http://www.shef.ac.uk/acse/research/ecrg/gat.html

Gen, M. and Cheng, R., 1997. Genetic Algorithms & Engineering Design, Wiley, New York.

Gerald, L., Gibson, P.E., 1997. A supervisory controller for optimization of building central cooling systems. ASHRAE Transactions. Symposia, PH-97-4-3, pp.486-493.

Giraud-Moreau, L. and Lafon, P., 2002. A comparison of evolutionary algorithms for mechanical design components. Engineering Optimization 34 pp.307-322.

Glover, F., 1989. Tabu search – Part I. ORSA Journal on Computing 1(3) pp.190-206.

Glover, F., 1990. Tabu search – Part II. ORSA Journal on Computing 2(1) pp.4-32.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley Publishing Company.

Goldfarb, D., and Lapidus, B., 1968. Conjugate gradient method for nonlinear programming problems with linear constraints. Industrial & Engineering Chemistry Fundamentals 7 pp.142-151.

Gouda, M.M., Danaher, S., Underwood, C.P., 2002. Building thermal model reduction using nonlinear constrained optimization. Building and Environment 37 pp.1255-1265.

Hadj-Alouance, A.B., Bean, J.C., 1997. A genetic algorithm for the multiple-choice integer program, Operations Research 45 pp.92-101.

Hanby, V.I., 2001. Evolutionary Gauss (real/integer) algorithm ver. 5.04 (Matlab), evolutionary algorithm developed with MATLAB, August 2001.

Hanby, V.I., 2002a. Evolutionary Gauss (real/integer) algorithm ver. 1.08 (multiple constraints), evolutionary algorithm developed with C++, February 2002.

Hanby, V.I., 2002b. Evolutionary Gauss (real/integer) algorithm ver. 5.10 (Matlab + TRNSYS), evolutionary algorithm developed with MATLAB, March 2002.

Hanby, V.I., and Angelov, P.P., 2000. Application of univariate search methods to the determination of HVAC plant capacity. Proc. CIBSE A: Building Services Engineering Research Technology 21(3) pp.161-166.

Hanby, V.I., Fletcher, D.W. and Jones, D.N.T., 2002. Optimal supervisory control of refrigeration plant by evolutionary strategy. Proceedings of System Simulation in Buildings '02: Liège, Belgium: University of Liège.

Hanby, V.I., Loveday, D.L. and Al-Ajmi, F., 2005. The optimal design for a ground cooling tube in a hot, arid climate. Building Services Engineering Research and Technology 26(1) pp.1-10.

Hanby, V.I. and Wright, J.A., 1989. HVAC optimization studies: Component modeling methodology. Building Services Engineering Research Technology 10(1) 35-39.

He, J. and Yao, X., 2003. Towards an analytic framework for analyzing the computation time of evolutionary algorithms. Artificial Intelligence 145 pp.59-97.

Hand, J.W., 2006. The ESP-r: cookbook, February 10, 2006.

Himmelblau, D.M., 1972. Applied Nonlinear Programming. McGraw-Hill, New York.

Hock, W. and Schittkowski, K., 1981. Test Examples for Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, Berlin, Germany.

Holland, J.H., 1962. Outline for a logical theory of adaptive systems. Journal of ACM 3 pp.297-314.

Holland, J.H., 1967. Nonlinear environments permitting efficient adaptation. Computer and Information Sciences II (New York: Academic).

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Harbor.

Homaifar, A., Lai, S.H.Y., Qi, X., 1994. Constrained optimization via genetic algorithms, Simulation 62(4) pp.242-254.

Hooke, R. and Jeeves, T.A., 1960. Direct search solution of numerical and statistical problems. Journal of Association of Computing Machinery 8 pp.212-229.

Huang, and Lam, 1997. Using genetic algorithms to optimize controller parameters for HVAC systems. Energy and Buildings 26 pp.277-282.

Huh, J., 1995. Optimal Air-Conditioning System Operating Strategies for Combined Temperature and Humidity Control in Buildings. PhD thesis, University of Colorado, US.

Iwamatsu, M., 2002. Generalized evolutionary programming with Lévy-type mutation. Computer Physics Communications 147 pp.729-732.

Ji, M., Tang, H., Gui, J., 2004. A single-point mutation evolutionary programming. Information Processing Letters 90 pp.293-299.

Johnson, E.G. and Abushagur, M.A.G., 1995. Micro-genetic algorithm optimization methods applied to dielectric gratings. Journal of the Optical Society of America 12(5) pp.1152-1160.

Joines, J., Houck, C., 1994. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs, in: D. Fogel (Ed.), Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE Press, Orlando, FL, pp.579-584.

Kannan, B.K., Kramer, S.N., 1994. An augmented Lagrange multiplier based method for mixed discrete continuous optimization and its applications to mechanical design, Journal of Mechanical Design Transaction ASME 116 pp.318-320.

Kao, C. and Chen, S., 2005. A stochastic quasi-Newton method for simulation response optimization. European Journal of Operational Research. (Article in press)

Kennedy, J. and Eberhart, R.C., 1995. Particle swarm optimization. In IEEE International Conference on Neural Networks, vol. IV, pp. 1942-1948, Perth, Australia, November 1995.

Kennett, S., 2001. Model Answer, Building Services Journal 23(9) pp.42-43.

Kintner-Meyer, M.C.W., 1994. An Investigation of Optimal Sizing Control of Air-Conditioning Systems in Commercial Buildings. PhD thesis, University of Washington, US.

Kirkpatrick, Jr.S., Gelatt, C., Vecchi, M., 1983. Optimization by simulated annealing. Science 220 (4598) pp.498-516.

Koeppel, E.A., Mitchell, J.W., Klein, S.A., Flake, B.A., 1995. Optimal supervisory control of an absorption chiller system. HVAC&R Research 1(4) pp.325-342.

Krishnakumar, K., 1989. Micro-genetic algorithms for stationary and non-stationary function optimization. SPIE Vol. 1196 Intelligent Control and Adaptive Systems.

Lambert, G.C. and Wright, J.A., 1991. Direct search optimization of HVAC systems design. Building Environmental Performance '91, BEPAC, University of Kent, Canterbury, UK, pp.196-206.

Lasdon, L.S., Warren, A.D., Jain, A., and Ratner, M., 1978. Design and testing of a generalized reduced gradient code for nonlinear programming. ACM Transaction. Mathematics Software 4 pp.34-50.

LBL and LANL 1982. DOE-2 Engineers Manual Version 2.1 A, Energy and Environment Division, Building Energy Simulation Group, Lawrence Berkeley Laboratory, and Group Q-11, Solar Energy Group, Energy Division, Los Alamos

National Laboratory, November 1982.

Lin, F. and Yi, J., 2000. Optimal operation of a CHP plant for space heating as a peak load regulating plant. Energy 25 pp.283-298.

Liu X., 2005. Global convergence on an active set SQP for inequality constrained optimization. Journal of Computational and Applied Mathematics 8(1) pp.201-211.

Lu L., Cai W., Chai Y.S., Xie, L., 2005a. Global optimization for overall HVAC systems – Part I problem formulation and analysis. Energy Conversion & Management 46 pp.999-1014.

Lu L., Cai W., Chai Y.S., Xie, L., 2005b. Global optimization for overall HVAC systems – Part II problem solution and simulations. Energy Conversion & Management 46 pp.1014-1028.

Lu L., Cai W., Soh Y.C., Xie L. and Li S., 2004. HVAC system optimization – condenser water loop. Energy Conversion and Management 45 pp.613-630.

Lu L., Cai W., Xie L., Li S., Soh Y., 2005c. HVAC system optimization – in-building section. Energy and Buildings 37 pp.11-22.

Maa, C. and Shanblatt, M., 1992. A two-phase optimization neural network. IEEE Transactions on Neural Networks 3(6) pp.003-1009.

Michalewicz, Z. and Attia, N.F., 1994. Evolutionary optimization of constrained problems, in: Proceedings of the 3rd Annual Conference on Evolutionary Programming, World Scientific, Singapore, pp.98-108.

Michalewicz, Z., Dasgupta, D., Le Riche, R.G. and Schoenauer, M., 1996. Evolutionary algorithms for constrained engineering problems. Computers Industrial Engineering 30(4) pp.851-870.

Michalewicz, Z. and Fogel, D.E., 2004. How to Solve It: Modern Heuristics, Springer, Heidelberg, 2004.

Michalewicz, Z, Nazhiyath, G and Michalewicz, M., 1996. A note on usefulness of geometrical crossover for numerical optimization problems. Evolutionary programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming ed L J Fogel, P J Angeline and T Bäck (Cambridge, Mass.: MIT Press) pp.305-312.

Nassif, N., Kajl, S., Sabourin, R., 2004. Two-objective on-line optimization of supervisory control strategy. Building Services Engineering Research and Technology 25(3) pp.241-251.

Nelder, J.A. and Mead, R. 1965. A simplex method for function minimization. The Computer Journal 7 pp.308-313.

Pohlheim, H., 2005. GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with Matlab. Internet web site: http://www.geatbx.com

Powell, D., Skolnick, M.M., 1993. Using generic algorithms in engineering design optimization with nonlinear constraints, in: S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kauffman, San Mateo, pp.424-430.

Powell, M.J.D., 1964. An efficiency way for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal 7 pp.155-162.

Ragsdell, K.M., Phillips, D.T., 1976. Optimal design of a class of welded structures using geometric programming, ASME J. Eng. Ind., Series B 98 (3) pp.1021-1025.

Rao, S.S., 1996. Engineering Optimization, third ed., Wiley, New York, 1996.

Rechenberg, I., 1965. Cybernetic Solution Path of an Experimental Problem. Royal Aircraft Establishment Library Transaction 1122.

Rechenberg, I., 1973. Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution, Frommann-Holzboog, Verlag, Stuttgart.

Reindl, D.T., Beckman, W.A. and Duffie, J.A., 1990. Evaluation of Hourly Tilted Surface Radiation Models, Solar Energy 45(1) pp.9-17.

Reklaitis, G.V., Ravindran, A., Ragsdell, K.M., 1983. Engineering Optimization Methods and Applications, Wiley, New York.

Ren, M.J. and Wright, J.A., 1998. A ventilated slab thermal storage system model. Building and Environment 33(1) pp.43-52.

Rink, R.E. and Li, N., 1995. Aggregation/disaggregation method for optimal control of multizone HVAC systems. Energy Conversion and Management 36(2) pp.79-86.

Runarsson, T.P. and Yao, X., 2000. Stochastic ranking for constrained evolutionary optimization. IEEE Transactions on Evolutionary Computation 4(3) pp.284-294.

Sakamoto, Y., Nagaiwa, A., Kobayasi, S. and Shinozaki, T., 1999. An optimization method of district heating and cooling plant operation based on genetic algorithm. ASHRAE Transactions 105(2) pp.1-11.

Sanaye S., Malekmohammadi, H.R., 2004. Thermal and economical optimization of air conditioning units with vapor compression refrigeration system. Applied Thermal Engineering 24 pp.1807-1825.

Schewfel H.-P., 1965. Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik Dipl.-Ing. Thesis, Technical University of Berlin, Hermann Föttinger Institute for Hydrodynamics.

Schewfel, H.-P., 1981. Numerical Optimization of Computer Models, Wiley, Great Britain, 1981.

van Schijndel, A.W.M., 2002. Optimal operation of a hospital power plant. Energy and Buildings 34 pp.1055-1065.

SEL 2000. TRNSYS A Transient System Simulation Program. Chapter 2, Volume I Reference Manual, Solar Energy Laboratory, University of Wisconsin-Madison, Madison, March 2000.

Senecal, P.K., 2000. Numerical optimization using the gen4 micro-genetic algorithm code. Engine Research Centre, University of Wisconsin-Madison, August 2000.

Senecal, P.K., Montgomery D.T. and Reitz, R.D., 2000. A methodology for engine design using multi-dimensional modeling and genetic algorithms with validation through experiments. International Journal of Engine Research 1 (3) pp.229-248.

Shanno, D.F., 1970. An accelerated gradient projection method for linearly constrained nonlinear estimation. SIAM Journal on Applied Mathematics 18, pp.322-334.

Siddall, J.N., 1972. Analytical Design-Making in Engineering Design, Prentice-Hall, Englewood Cliffs, NJ.

Silverman, G.J., Jurovics, S.A., Low, D.W., Sowell, E.F., 1981. Modeling and optimization of HVAC systems using network concepts. ASHRAE Transactions 87(2) pp.585-597.

Simpson, A.R., Dandy, G.C. and Murphy, L.J., 1994. Genetic algorithms compared to other techniques for pipe optimization. Journal of Water Resources Planning and Management 120 (4) pp.423-443.

Sowell, E.F., 1990. A numerical lighting/HVAC test cell. ASHRAE Transactions 96(2) pp.780-786.

Söylemez, M.S., 2001. On the optimum channel sizing for HVAC systems. Energy Conversion and Management 42 pp.791-798.

SPARE 2002. Study on the Potential Applications of Renewable Energy in Hong Kong, Stage 1 Study Report, Electrical & Mechanical Services Department, Government of the Hong Kong Special Administrative Region, December 2002.

Taylor, R.D., 1996. Development of an Integrated Building Energy Simulation with Optimal Central Plant Control. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA.

TESS 2004. TESS Library Documentation, TESS Component Libraries v2.0 for TRNSYS v16.x and TRNSYS Simulation Studio, Thermal Energy System Specialists.

Threlkeld, J.L., 1970. Thermal Environmental Engineering, Prentice-Hall, New York, Second Edition.

Tsal, R.J., Behls, H.F. and Mangel, R., 1988. T-method duct design, Part I: Optimization theory; Part II: Calculation procedure and economic analysis. ASHRAE Transactions 94(2) pp.90-111.

Wang, S. and Wang, J., 2002. Robust sensor fault diagnosis and validation in HVAC systems. Transactions of the Institute of Measurement and Control 24(3) pp.231-262.

Wen, F. and Chang, C.S., 1997. Transmission network optimal planning using the tabu search method. Electric Power System Research 42 pp.153-163.

Wetter, M., 2001. GenOpt – a generic optimization program. Proceedings of Building Simulation 2001, Seventh International IBPSA Conference, Rio de Janeiro, Brazil.

Wetter, M., 2004. GenOpt - Generic Optimization Program: User Manual Version 2.0.0. January 5, 2004. Lawrence Berkeley National Laboratory.

Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1) pp.67-82.

Wright, J.A., 1986. The Optimised Design of HVAC Systems. PhD thesis, Loughborough University of Technology, UK.

Wright, J.A., 1996. HVAC optimisation studies: sizing by genetic algorithm. Building Services Engineering Research Technology 17(1) pp.7-14.

Wright, J.A. and Farmani, R., 2001. Genetic algorithms: a fitness formulation for constrained minimization. GECCO-2001: Proceedings of the Genetic and Evolutionary Computation Conference July 2001, pp.725-732.

Wright, J.A. and Hanby, V.I., 1987. The formulation, characteristics, and solution of HVAC system optimized design problems. ASHRAE Transaction 93(2) pp.2133-2145.

Wright, J.A., Loosemore, H.A., Farmani, R., 2002. Optimization of building thermal design and control by multi-criterion genetic algorithm. Energy and Buildings 34 pp.959-972.

Xiao Fengchao and Yabe Hatsuo, 1998. Microwave imaging of perfectly conducting cylinders from real data by micro genetic algorithm coupled with deterministic method. IEICE Transactions on Electronics, E81-C(12) pp.1784-1792, December 1998.

Yang, I.H., Yeo, M.S., Kim, K.W., 2003. Application of artificial neural network to predict the optimal start time for heating system in building. Energy Conversion and Management 44 pp.2791-2809.

Yao, X., Liu, Y., Lin, G., 1999. Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation 3(2) pp.82-102.

Yik, F.W.H. and Burnett, J., 1998. BECON: A program for predicting building energy consumption for air-conditioning buildings. Transactions HKIE 5(3) pp.89-94.

Yik, F.W.H. and Chan, T.K., 1998. Air-conditioning system modeling in BECON. Proceedings, CIBSE National Conference, Bournemouth pp.40-49.

Youssef, H., Sait, S.M., Adiche, H., 2001. Evolutionary algorithms, simulated annealing and tabu search: a comparative study. Engineering Applications of Artificial Intelligence 14 pp.167-181.

Yuen, S.Y. and Cheung, B.K.S., 2006. Bounds for probability of success of classical genetic algorithm based on Hamming distance. IEEE Transactions on Evolutionary Computation 10(1) pp.1-18, February 2006.

Zaheer-Uddin, M., Zheng, G.R., 2000. Optimal control of time-scheduled heating, ventilating and air conditioning processes in buildings. Energy Conversion and Management 41 pp.49-60.

Zheng, G.R. and Zaheer-Uddin, M., 1996. Optimization of thermal process in a variable air volume HVAC system. Energy 21(5) pp.407-420.

APPENDIX DETAILS OF TEST FUNCTIONS AND TEST PROBLEMS

Table A1 Summary of characteristics of test functions and test problems

Test function	n_{var}	n_{con}	Type of function	No. of linear inequalities	No. of non-linear equalities	No. of non-linear inequalities	No. of active constraints
f_{c1} (Floundas & P)	13	9	quadratic	9	0	0	6
f_{c2} (Himmelblau 11)	5	6	quadratic	0	0	6	2
f_{c3} (Floundas & P)	2	2	cubic	0	0	2	2
f_{c4} (Hock & S 113)	10	8	quadratic	3	0	5	6
f_{c5} (Hock & S 100)	7	4	polynomial	0	0	4	2
f_{c6} (Hock & S HX)	8	6	linear	3	0	3	6
f_{c7} (Maa & Shanblatt)	2	2	quadratic	0	1	0	1
f_{c8} (Deb 1)	2	2	polynomial	0	0	2	---
f_{c9} (Hock & S 85)	5	38	polynomial	3	0	35	---
f_{c10} (Welded Beam 1)	4	5	polynomial	1	0	4	---
f_{c11} (Welded Beam 2)	4	7	polynomial	2	0	5	---
f_{c12} (Pressure Vessel)	4	4	polynomial	3	0	1	---
f_{u1} (Sphere Model)	10	---	quadratic	---	---	---	---
f_{u2} (Schwefel 2.22)	10	---	polynomial	---	---	---	---
f_{u3} (Schwefel 1.2)	10	---	quadratic	---	---	---	---
f_{u4} (Rosenbrock)	10	---	polynomial	---	---	---	---
f_{u5} (Easom)	2	---	trigonometric & exponential	---	---	---	---
f_{m1} (Schwefel 7)	10	---	trigonometric	---	---	---	---
f_{m2} (Rastrigin)	10	---	quadratic & trigonometric	---	---	---	---
f_{m3} (Ackley)	10	---	exponential & trigonometric	---	---	---	---
f_{m4} (Griewank)	10	---	quadratic & trigonometric	---	---	---	---
f_{m5} (Senecal)	2	---	exponential & trigonometric	---	---	---	---
f_{m6} (De Jong SMF)	2	---	non-quadratic	---	---	---	---

Test Function f_{c1} – Floundas and Pardalos' Problem (Floundas and Pardalos 1987)

Objective Function: $$f_{c1}(x) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \quad \text{(A1)}$$

LI Constraint Function:
$10 - 2x_1 - 2x_2 - x_{10} - x_{11} \geq 0$ (active)
$10 - 2x_1 - 2x_3 - x_{10} - x_{12} \geq 0$ (active)
$10 - 2x_2 - 2x_3 - x_{11} - x_{12} \geq 0$ (active)
$8x_1 - x_{10} \geq 0$
$8x_2 - x_{11} \geq 0$
$8x_3 - x_{12} \geq 0$
$2x_4 + x_5 - x_{10} \geq 0$ (active)
$2x_6 + x_7 - x_{11} \geq 0$ (active)
$2x_8 + x_9 - x_{12} \geq 0$ (active)

for $0 \leq x_i \leq 1$, i = 1, 2, ..., 9
$0 \leq x_i \leq 100$, i = 10, 11, 12
$0 \leq x_{13} \leq 1$

Optimum (minimum): $\mathbf{x^*} = (1,1,1,1,1,1,1,1,1,3,3,3,1)$
$f_{c1}(\mathbf{x^*}) = -15$

Test Function f_{c2} – Himmelblau's Problem 11

Objective Function: $$f_{c2}(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \quad \text{(A2)}$$

NI Constraint Function:
$92 - 85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \geq 0$ (active)
$85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \geq 0$
$110 - 80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 \geq 0$
$80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 90 \geq 0$
$25 - 9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 \geq 0$
$9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 20 \geq 0$ (active)

for $78 \leq x_1 \leq 102$
$33 \leq x_2 \leq 45$
$27 \leq x_i \leq 45$, i = 3, 4, 5

Optimum (minimum): $\mathbf{x^*} = (78.0, 33.0, 29.995, 45.0, 36.776)$
$f_{c2}(\mathbf{x^*}) = -30665.5$

Test Function f_{c3} –Floundas and Pardalos' Problem (Floundas and Pardalos 1987)

Objective Function: $$f_{c3}(x) = (x_1 - 10)^3 + (x_2 - 20)^3 \quad \text{(A3)}$$

NI Constraint Function:
$(x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geq 0$ (active)
$-(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 \geq 0$ (active)

for $13 \leq x_1 \leq 100$
$0 \leq x_2 \leq 100$

Optimum (minimum): $\mathbf{x^*} = (14.095, 0.84296)$
$f_{c3}(\mathbf{x^*}) = -6961.81381$

Test Function f_{c4} – Hock and Schittkowski's *Problem 113* (Hock and Schittkowski 1981)

Objective Function:
$$f_{c4}(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \quad \text{(A4)}$$

LI Constraint Function:
$$105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0 \text{ (active)}$$
$$-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0 \text{ (active)}$$
$$-10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0 \text{ (active)}$$

NI Constraint Function:
$$-x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geq 0 \text{ (active)}$$
$$8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0 \text{ (active)}$$
$$-5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geq 0 \text{ (active)}$$
$$3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 0$$
$$-0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geq 0$$

for $-10.0 \leq x_i \leq 10.0$, $i = 1, 2, \ldots, 10$

Optimum (minimum):
$$\mathbf{x}^* = (2.1711996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$$
$$f_{c4}(\mathbf{x}^*) = 24.3062091$$

Test Function f_{c5} – Hock and Schittkowski's Problem 100 (Hock and Schittkowski 1981)

Objective Function:
$$f_{c5}(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \quad \text{(A5)}$$

NI Constraint Function:
$$127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0 \text{ (active)}$$
$$282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geq 0$$
$$196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geq 0$$
$$-4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0 \text{ (active)}$$

for $-10.0 \leq x_i \leq 10.0$, $i = 1, 2, \ldots, 7$

Optimum (minimum):
$$\mathbf{x}^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$$
$$f_{c5}(\mathbf{x}^*) = 680.6300573$$

Test Function f_{c6} – Hock and Schittkowski's Heat Exchanger Design (Hock and Schittkowski 1981)

Objective Function: $f_{c6}(x) = x_1 + x_2 + x_3$ (A6)

LI Constraint Function:
$1 - 0.0025\,(x_4 + x_6) \geq 0$ (active)
$1 - 0.0025\,(x_5 + x_7 - x_4) \geq 0$ (active)
$1 - 0.01\,(x_8 - x_5) \geq 0$ (active)

NI Constraint Function:
$x_1x_6 - 833.33252x_4 - 100x_1 + 83333.333 \geq 0$ (active)
$x_2x_7 - 1250x_5 - x_2x_4 + 1250x_4 \geq 0$ (active)
$x_3x_8 - 1250000 - x_3x_5 + 2500x_5 \geq 0$ (active)

for $100 \leq x_1 \leq 10000$
$1000 \leq x_i \leq 10000,\ i = 2, 3$
$10 \leq x_i \leq 1000,\ i = 4, 5, \ldots, 8$

Optimum (minimum): $\mathbf{x}^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)$
$f_{c6}(\mathbf{x}^*) = 7049.330923$

Test Function f_{c7} – Maa and Shanblatt's Problem (Maa and Shanblatt 1992)

Objective Function: $f_{c7}(x) = x_1^2 + (x_2 - 1)^2$ (A7)

NE Constraint Function: $x_2 - x_1^2 = 0$

for $-1 \leq x_i \leq 1,\ i = 1, 2$

Optimum (minimum): $\mathbf{x}^* = (\pm 0.70711, 0.5)$
$f_{c7}(\mathbf{x}^*) = 0.75000455$

Test Function f_{c8} – Deb's Test Problem 1 (Deb 2000)

Objective Function: $f_{c8} = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$ (A8)

NI Constraint Function:
$4.84 - (x_1 - 0.05)^2 - (x_2 - 2.5)^2 \geq 0$
$x_1^2 + (x_2 - 2.5)^2 - 4.84 \geq 0$

for $0 \leq x_1, x_2 \leq 6$

Optimum (minimum): $\mathbf{x}^* = (2.246826, 2.381865)$
$f_{c8}(\mathbf{x}^*) = 13.59085$

Test Function f_{c9} – Hock & Schittkowski's Problem 85 (Hock and Schittkowski 1981)

Objective Function:
$$f_{c9} = -0.1365 - 5.843\times10^{-7}y_{17} + 1.17\times10^{-4}y_{14} + 2.358\times10^{-5}y_{13} + 1.502\times10^{-6}y_{16} + 0.0321y_{12} + 0.004324y_5 + 1\times10^{-4}c_{15}/c_{16} + 37.48\,y_2/c_{12} \quad \text{(A9)}$$

LI Constraint Function:
$$1.5x_2 - x_3 \geq 0$$
$$y_1 - 213.1 \geq 0$$
$$405.23 - y_1 \geq 0$$

NI Constraint Function:
$$y_j - a_j \geq 0 \qquad j = 2,\ldots, 17$$
$$b_j - y_j \geq 0 \qquad j = 2,\ldots, 17$$
$$y_4 - (0.28/0.72)y_5 \geq 0$$
$$21 - 3496y_2/c_{12} \geq 0$$
$$62212/c_{17} - 110.6 - y_1 \geq 0$$

where,
$$y_1 = x_2 + x_3 + 41.6$$
$$c_1 = 0.024x_4 - 4.62$$
$$y_2 = 12.5/c_1 + 12$$
$$c_2 = 0.0003535x_1x_1 + 0.5311x_1 + 0.08705y_2x_1$$
$$c_3 = 0.052x_1 + 78 + 0.002377y_2x_1$$
$$y_3 = c_2/c_3$$
$$y_4 = 19y_3$$
$$c_4 = 0.04782(x_1 - y_3) + 0.1956(x_1 - y_3)^2/x_2 + 0.6376y_4 + 1.594y_3$$
$$c_5 = 100x_2$$
$$c_6 = x_1 - y_3 - y_4$$
$$c_7 = 0.95 - c_4/c_5$$
$$y_5 = c_6c_7$$
$$y_6 = x_1 - y_5 - y_4 - y_3$$
$$c_8 = 0.995(y_4 + y_5)$$
$$y_7 = c_8/y_1$$
$$y_8 = c_8/3798$$
$$c_9 = y_7 - 0.0663y_7/y_8 - 0.3153$$
$$y_9 = 96.82/c_9 + 0.321y_1$$
$$y_{10} = 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6$$
$$y_{11} = 1.71x_1 - 0.452y_4 + 0.58y_3$$
$$c_{10} = 12.3/752.3$$
$$c_{11} = 1.75y_20.995x_1$$
$$c_{12} = 0.995y_{10} + 1998$$
$$y_{12} = c_{10}x_1 + c_{11}/c_{12}$$
$$y_{13} = c_{12} - 1.75y_2$$
$$y_{14} = 3623 + 64.4x_2 + 58.4x_3 + 146312/(y_9 + x_5)$$
$$c_{13} = 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095$$
$$y_{15} = y_{13}/c_{13}$$
$$y_{16} = 148000 - 331000y_{15} + 40y_{13} - 61y_{15}y_{13}$$
$$c_{14} = 2324y_{10} - 28740000y_2$$
$$y_{17} = 14130000 - 1328y_{10} - 531y_{11} + c_{14}/c_{12}$$
$$c_{15} = y_{13}/y_{15} - y_{13}/0.52$$
$$c_{16} = 1.104 - 0.72y_{15}$$
$$c_{17} = y_9 + x_5$$

a[i] = {0, 17.505, 11.275, 214.228, 7.458, 0.961, 1.612, 0.146, 107.99, 922.693, 926.832, 18.766, 1072.163, 8961.448, 0.063, 71084.33, 2802713}

b[i] = {0, 1053.6667, 35.03, 665.585, 584.463, 265.916, 7.046, 0.222, 273.366, 1286.105, 1444.046, 537.141, 3247.039, 26844.086, 0.386, 140000, 12146108}, for i = 1,..., 17

for $704.4148 \le x_1 \le 906.3855$
$68.6 \le x_2 \le 288.88$
$0 \le x_3 \le 134.75$
$193 \le x_4 \le 287.0966$
$25 \le x_5 \le 84.1988$

Optimum (minimum): $\mathbf{x}^* = (707.337769, 68.600273, 102.900146, 282.024841, 84.198792)$
$f_{c9}(\mathbf{x}^*) = -1.91460$

Test Function f_{c10} – Welded Beam Design Problem 1 (Reklaitis *et al.* 1983)

Objective Function: $f_{c10} = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2)$ (A10)

LI Constraint Function: $x_4 - x_1 \ge 0$

NI Constraint Function: $13600 - \tau \ge 0$
$30000 - \sigma \ge 0$
$P_c - 6000 \ge 0$
$0.25 - \delta \ge 0$

where, $\tau = \sqrt{(\tau')^2 + (\tau'')^2 + x_2\tau'\tau''/\sqrt{0.25[x_2^2 + (x_1 + x_3)^2]}}$
$\sigma = 504000 / (x_3^2x_4)$
$P_c = 64746.022(1 - 0.0282346x_3)x_3x_4^3$
$\delta = 2.1952 / (x_3^3x_4)$
$\tau' = 6000 / (\sqrt{2}\, x_1x_2)$
$$\tau'' = \frac{6000(14 + 0.5x_2)\sqrt{0.25[x_2^2 + (x_1 + x_3)^2]}}{2\{0.707x_1x_2[x_2^2/12 + 0.25(x_1 + x_3)^2]\}}$$

for $0.125 \le x_1 \le 10$
$0.1 \le x_i \le 10$, for i= 2, 3, 4

Optimum (minimum): $\mathbf{x}^* = (0.2444, 6.2187, 8.2915, 0.2444)$
$f_{c10}(\mathbf{x}^*) = 2.38116$

Test Function f_{c11} – Welded Beam Design Problem 2 (Rao 1996)

Objective Function: $f_{c11} = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2)$ (A11)

LI Constraint Function:
$x_4 - x_1 \geq 0$
$x_1 - 0.125 \geq 0$

NI Constraint Function:
$13600 - \tau \geq 0$
$30000 - \sigma \geq 0$
$P_c - 6000 \geq 0$
$0.25 - \delta \geq 0$
$5 - 0.10471x_1^2 - 0.04811x_3x_4(14 + x_2) \geq 0$

where, $\tau = \sqrt{(\tau')^2 + (2\tau'\tau''x_2/2R) + (\tau'')^2}$

$\tau' = P/(\sqrt{2}\,x_1x_2)$

$\tau'' = MR/J$

$M = P(L + x_2/2)$

$R = \sqrt{(x_2^2/4) + [(x_1 + x_3)/2]^2}$

$J = 2\{\sqrt{2}\,x_1x_2[(x_2^2/12) + ((x_1 + x_3)/2)^2]\}$

$\sigma = 6PL/(x_4x_3^2)$

$\delta = 4PL^3/(Ex_3^3x_4)$

$$P_c = \frac{4.013E\sqrt{x_3^2x_4^6/36}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$

$P = 6000$, $L = 14$, $E = 30\times10^6$, $G = 12\times10^6$

for $0.125 \leq x_1 \leq 10$
$0.1 \leq x_i \leq 10$, for $i = 2, 3, 4$

Optimum (minimum): $\mathbf{x}^* = (0.2088, 3.4205, 8.9975, 0.2100)$
$f_{c11}(\mathbf{x}^*) = 1.74830941$

Test Function f_{c12} – Pressure Vessel Design Problem (Kannan and Kramer 1994)

Objective Function: $f_{c12} = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3)$ (A12)

LI Constraint Function:
$-0.0193x_3 + x_1 \geq 0$
$-0.00954x_3 + x_2 \geq 0$
$240 - x_4 \geq 0$

NI Constraint Function: $\pi x_3^2x_4 + 4\pi\, x_3^3/3 - 1296000 \geq 0$

for $1 \leq x_1, x_2 \leq 20$
$0 \leq x_3 \leq 100$
$100 \leq x_4 \leq 200$

Optimum (minimum): $\mathbf{x}^* = (0.8125, 0.4375, 40.3239, 200)$
$f_{c12}(\mathbf{x}^*) = 6288.7445$

Test Function f_{u1} – Sphere Model

Objective Function: $f_{u1}(x) = \sum_{i=1}^{n} x_i^2$ (A13)

for $-100 \le x_i \le 100$

Optimum (minimum): $x^* = (0, 0, \ldots, 0)$
$f_{u1}(x^*) = 0$

Test Function f_{u2} – Schwefel's Problem 2.22

Objective Function: $f_{u2}(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$ (A14)

for $-10 \le x_i \le 10$

Optimum (minimum): $x^* = (0, 0, \ldots, 0)$
$f_{u2}(x^*) = 0$

Test Function f_{u3} – Schwefel's Problem 1.2

Objective Function: $f_{u3}(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$ (A15)

for $-100 \le x_i \le 100$

Optimum (minimum): $x^* = (0, 0, \ldots, 0)$
$f_{u3}(x^*) = 0$

Test Function f_{u4} – Generalized Rosenbrock's Function

Objective Function: $f_{u4}(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$ (A16)

for $-30 \le x_i \le 30$

Optimum (minimum): $x^* = (1, 1, \ldots, 1)$
$f_{u4}(x^*) = 0$

Test Function f_{u5} – Easom's Function

Objective Function: $f_{u5}(x) = -\cos x_1 \cos x_2 \exp\{-[(x_1 - \pi)^2 + (x_2 - \pi)^2]\}$ (A17)
for $-100 \le x_i \le 100$

Optimum (minimum): $x^* = (\pi, \pi)$
$f_{u5}(x^*) = -1$

Test Function f_{m1} – Schwefel's Function 7

Objective Function: $$f_{m1}(x) = \sum_{i=1}^{n}[-x_i \sin(\sqrt{|x_i|})] \quad \text{for} \quad -500 \le x_i \le 500 \tag{A18}$$

Optimum (minimum): $x^* = (420.9687, 420.9687, \ldots, 420.9687)$
$f_{m1}(x^*) = -n_{var} \cdot 418.9829$

Test Function f_{m2} – Generalized Rastrigin's Function

Objective Function: $$f_{u7}(x) = \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10] \quad \text{for} \quad -5.12 \le x_i \le 5.12 \tag{A19}$$

Optimum (minimum): $x^* = (0, 0, \ldots, 0)$
$f_{m2}(x^*) = 0$

Test Function f_{m3} – Ackley's Function

Objective Function: $$f_{m3}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left[\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right] + 20 + e \quad \text{for} \quad -32 \le x_i \le 32 \tag{A20}$$

Optimum (minimum): $x^* = (0, 0, \ldots, 0)$
$f_{m3}(x^*) = 0$

Test Function f_{m4} – Generalized Griewank Function

Objective Function: $$f_{m4}(x) = \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad \text{for} \quad -600 \le x_i \le 600 \tag{A21}$$

Optimum (minimum): $x^* = (0, 0, \ldots, 0)$
$f_{m4}(x^*) = 0$

Test Function f_{m5} – Senecal's Example Problem (Senecal 2000)

Objective Function: $f_{m5}(x) = f_1 f_2 f_3 f_4$ (A22)

where,
$f_1 = [\sin(5.1\pi x_1 + 0.5)]^6$
$f_2 = \exp[-4 \ln(2) (x_1 - 0.0667)^2 / 0.64]$
$f_3 = [\sin(5.1\pi x_2 + 0.5)]^6$
$f_4 = \exp[-4 \ln(2) (x_2 - 0.0667)^2 / 0.64]$

for $0 \le x_i \le 1$

Optimum (maximum): $\mathbf{x}^* = (0.0667, 0.0667)$
$f_{m5}(\mathbf{x}^*) = 1$

Test Function f_{m6} – De Jong's Stationary Multimodal Function (De Jong 1975)

Objective Function:
$$f_{m6}(x) = \frac{1}{0.002 + \sum_{j=1}^{25} \frac{1}{f_j(x)}}$$ (A23)

where,
$$f_j(x) = j + \sum_{i=1}^{2} (x_i - a_{ij})^6$$

for $-65.536 \le x_i \le 65.536$

Optimum (minimum): $\mathbf{x}^* = (1, 1)$ for $a_{ij} = 1$
$f_{m6}(\mathbf{x}^*) = 0.2619201$